Literals與Clauses:

實際上的literal是帶正負值(positive/negative)得，而clauses是好幾個literals組合而成的。如果我自訂一兩個Class來表示這兩個概念，會因為要自己補\_\_eq\_\_和\_\_hash\_\_而變得很麻煩，所以我直接用整數來表達，正數表示positive literal，負數表示negative literal，而clauses就是一個set of literals。例如：{1, 2, 3}表示一個clause，其中包含三個literals，分別是1, 2, 3。而因為set是python內建的資料結構，所以用起來會比自定義的class方便很多。而在這個作業裡面，我把KB定義為一個list of clauses，例如：[{1, 2, 3}, {1, 2, 4}]，KB0則因為都是single literal，所以就是單純整數的set，例如：{1, -2, -3, 4}，表示1, 4是Mine，2, 3是Safe。

執行流程:

game\_start(): 一直跑play()這個function，並在跑不下去(stuck)或是過關的時候幫忙確認玩家的答案是否正確。

generate\_mine(): 產生地雷的位置，回傳一個list of tuple，每個tuple是一個地雷的位置。

generate\_board(): 根據是player還是answer，產生對應的空白地圖。

assign\_answer\_board(): 根據地雷的位置，把地雷的位置填入地圖，並且計算每個格子周圍的地雷數量。

get\_init\_safe(): 隨機產生一個safe的位置，並且回傳。

print\_board(): 印出player或是answer的地圖。

dig(): 根據玩家輸入的位置，挖掉該位置，如果挖到地雷就回傳-1，否則回傳該位置周圍的地雷數量(可以從answer\_board得到)。

clean\_KBs(): 把KB裡面的所有clauses都掃描一遍，如果裡面已經有確認過的literals，就把該clauses刪掉，代表這個clauses已經沒有用了(已經恆成立了)，然後取unique和根據長度排序，這個function只是為了確保KB的資料是最精簡的狀態。

KB\_to\_KB0(): id是傳入值去掉正負號，mine\_or\_safe = tar > 0，如果mine\_or\_safe是True，代表是Mine，否則是Safe。因為不能一邊迭代一邊刪除，所以先把要刪除的index記下來，最後再一次刪除。而如果id一樣，要看是不是同號，如果同號就把整個clauses從KB刪除，如果不同號就把該literal從clauses刪除。再來確認KB內有沒有clause是其他clauses的subset，如果有就把super set刪除，直到沒有新的subset出現為止。

add\_to\_KB(): 如果KB裡面已經有這個clauses了，就不用加入了，如果沒有而且長度等於1，就把這個clauses直接加入KB。剩下的把負面敘述出現在KB0裡面的literal刪掉，同時做subsumption直到沒有新的subset出現為止。然後再把自己加到KB裡面。

play(): 先執行一次clean\_KBs()，確保KB和KB0的資料都是最精簡的狀態，先找一個在KB裡面的single literal當成目標，如果找不到救回傳-1，然後把這個目標透過KB\_to\_KB0()加入KB0，如果目標是Mine，就把他mark起來，如果是Safe，就dig它，如果回傳-1，代表挖到地雷，就回傳-1，否則看周圍那些座標(around)，如果dig的回傳是0，所有周圍的地都是安全的，就直接把around裡面的座標做成single negative literals加到KB中，如果周圍全部都是地雷，則把around裡面的座標做成single positive literals加到KB中，如果周圍有地雷也有安全的，就利用排列組合的工具，把周圍的座標做成all positive/negative clauses加到KB中，然後再執行一次clean\_KBs()，確保KB和KB0的資料都是最精簡的狀態，然後就return回去，繼續下一輪的play()，直到KB裡面沒有clauses了。

check\_win(): 檢查玩家的地圖和答案是否一樣，如果除了地雷的位置的標誌不同以外，其他都一樣，就代表過關了，回傳True，否則回傳False。

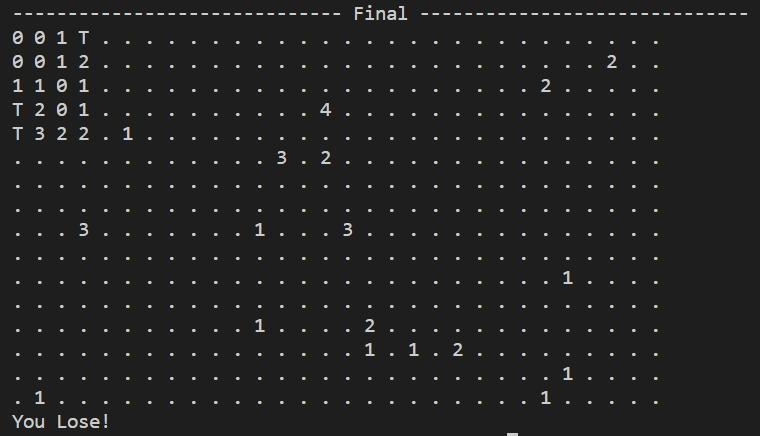
一些實驗與發現:

我自己在測試的時候，發現Hard的場地其實基本上很難過，因為地雷的數量太多了，基本上快要是Easy的兩倍，這樣地雷的比例之下就算是給兩倍的多比例的init\_safe也很難過關。還有不管在甚麼難度之下，只要有出現一整排的數字，會因為可行種類太多而卡住，然後就過不了，也可能有地雷直接圍成一圈，直接導致無法判斷，這些都是我在測試的時候發現的問題。下面幾張圖是一些範例:

一張含有 文字, 電子產品, 鍵盤 的圖片

自動產生的描述

數字圍成一圈，太多種合法可能，所以stuck



數字太過分散，起始點抽到的0太少，KB的single literal太少所以stuck

一張含有 文字, 時鐘, 測量儀器, 手錶 的圖片

自動產生的描述

被地雷圍繞一整圈，根本沒辦法知道任何有關中間那點的資訊，這也會stuck。但如果有設定好地雷的總數，這個情況是可以被解出來的。

Code: <https://gist.github.com/ypwang0408/53edb877c6c87dc096fa1e23d3e5c487>