**期中報告**

**走迷宮範例 – 使用Python**

走迷宮一直是一個很有趣的例子，以下是用Python實作的走迷宮範例

首先做一個6\*6大小的迷宮

#maze = [[2,2,2,2,2,2],  
               [2,0,0,0,2,2],  
               [2,0,2,0,0,2],  
               [2,0,0,2,0,2],  
               [2,0,2,2,0,2],  
               [2,2,2,2,2,2]]

# 向上走的函數  
def goUp(location):  
#     如果位置已在最上面那排，則不需要向上走  
     if(location[1]== 0):  
         return False  
     else:  
#         如果不是在最上排，則向上走  
         newLocation = [location[0], location[1]-1]  
#         如果已經走過，就回傳False  
        if newLocation in routeHistory:  
             return False  
#         如果這個位置是牆，則回傳False  
         elif maze[newLocation[0]][newLocation[1]] == 2:  
             return False  
#         如果這個位置是正確的話，則回傳True  
         else:  
             rightRoute.append(newLocation)  
             routeHistory.append(newLocation)  
             return True

#         接下來下面3個Function其實都一樣，只是方向與上方不同(分別是下、左、右)

def goDown(location):  
     if(location[1]==5):  
         return False  
     else:  
         newLocation = [location[0], location[1]+1]  
         if newLocation in routeHistory:  
             return False         
         elif maze[newLocation[0]][newLocation[1]] == 2:  
             return False  
         else:  
             rightRoute.append(newLocation)  
             routeHistory.append(newLocation)  
             return True

def goLeft(location):  
     if(location[0]==0):  
         return False  
     else:  
         newLocation = [location[0]-1, location[1]]  
         if newLocation in routeHistory:  
             return False         
         elif maze[newLocation[0]][newLocation[1]] == 2:  
             return False  
         else:  
             rightRoute.append(newLocation)  
             routeHistory.append(newLocation)  
             return True  
          
def goRight(location):  
     if(location[0]==5):  
         return False  
     else:  
         newLocation = [location[0]+1, location[1]]  
         if newLocation in routeHistory:  
             return False         
         elif maze[newLocation[0]][newLocation[1]] == 2:  
             return False  
         else:  
             rightRoute.append(newLocation)  
             routeHistory.append(newLocation)  
             return True  
              
# 這個List用來記錄正確路徑  
rightRoute = [[1,1]]  
# 記錄已經走過的路徑  
routeHistory = [[1,1]]  
# 現在的位置  
location = [1,1]

# 如果最後一個位置不是在出口[4,4]，則繼續搜尋  
while rightRoute[-1] != [4,4]:  
#     以下分別有4個函數，分別代表要向上下左右的方向走，執行函數後，如果回傳True則用下一個位置繼續走，如果回傳False則換方向走  
    if goUp(location):  
         location = rightRoute[-1]  
         continue  
      
     if goDown(location):  
         location = rightRoute[-1]  
         continue  
          
     if goLeft(location):  
         location = rightRoute[-1]  
         continue  
          
     if goRight(location):  
         location = rightRoute[-1]  
         continue  
#     如果有函數回傳False，代表這個位置沒路走，則把這個位置從正確路徑中剔除  
     rightRoute.pop()  
#     剔除後重新用新的位置繼續尋找  
     location = rightRoute[-1]  
# 最後印出路徑  
print(rightRoute)

參考網址<https://newaurora.pixnet.net/blog/post/224819033-%E8%B5%B0%E8%BF%B7%E5%AE%AE%E7%AF%84%E4%BE%8B---%E4%BD%BF%E7%94%A8python>