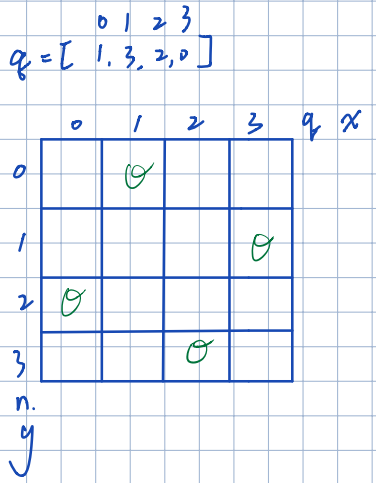
八皇后問題：如何在8*8的西洋棋棋盤上放置八個皇后，使任何一個皇后都無法直接吃掉其他皇后？
為了達到此目的，任兩個皇后都不能處於同一條橫行、縱行或斜線上。



queen.py

```
# 資料結構
#   四皇后解答 q=[1,3,0,2] 代表
#       x=[1,3,0,2]
#       y=[0,1,2,3]  y固定
def queen(n): # n 皇后主函數
    q = [] # q 代表已經排下去的皇后，一開始還沒排，所以是空的
    return queenNext(n, q) # 呼叫 queenNext 遞迴下去排出所有可能

# 思考：排到一半 q=[1,3] 繼續排下去
#     對 [1,3,0..], [1,3,1..], [1,3,2..], [1,3,3..] 全部試一遍
def queenNext(n, q): # 已經排好 q[0..y2-1], 繼續排下去 [y2...n-1]
    y2 = qlen = len(q)
    if qlen == n: # 全部排好了
        print(q) # 印出盤面   假設q只有一個元素，代表第0排排好了
        return                假設q只有二個元素，代表第0、1排排好了
    # 還沒排滿，繼續排下去
    for x2 in range(n): # 對本列的每一個 x2 去嘗試   第2排的x
        isConflict = False
        for y in range(qlen): # 前面已經排下去的舊皇后，座標為 (x,y)
            x = q[y]
            if conflict(x,y,x2,y2): # 檢查新排的皇后(x2,y2)，和前面的有沒有衝突
                isConflict = True
        if not isConflict:  # 如果沒有衝突，就繼續排下去
            q.append(x2)    # 把 (x2,y2) 放進盤面
            queenNext(n, q) # 繼續遞迴尋找下一個皇后位置
            q.pop()         # 把 (x2,y2) 移出盤面

def conflict(x1,y1,x2,y2): # 判斷 (x1,y1), (x2,y2) 兩個位置有沒有衝突
    if x1==x2: return True
    if y1==y2: return True
    if x1+y1==x2+y2: return True
    if x1-y1==x2-y2: return True
    return False

queen(4) # 列出四皇后的解答
queen(8) # 列出八皇后的解答
```

前導問題 - 排列

permutation.py

0,0,0
0,0,1
0,0,2
0,1,0
0,1,1

```
def perm(n): # 主函數
    p = [] # p 代表已經排下去的，一開始還沒排，所以是空的
    return permNext(n, p) # 呼叫 permNext 遞迴下去排出所有可能

# 思考：排到一半 p=[1,3] 繼續排下去
def permNext(n, p): # 已經排好 p[0..i-1], 繼續排下去 [i...n-1]
    i = len(p)
    if i == n:  # 全部排好了
        print(p)  # 印出排列
        return
    # 還沒排滿，繼續排下去
    for x in range(n): # 對本列的每一個 x 去嘗試
        p.append(x)    # 把 x 放進盤面
        permNext(n, p) # 繼續遞迴尋找下一個排列
        p.pop()        # 把 x 移出盤面

perm(2) # 列出 2 個的排列
perm(3) # 列出 3 個的排列
```

真值表

truthtable.py

```
def truthTable(n): # 列出 n 變數的所有可能 0,1 排列
    p = [] # p 代表已經排下去的，一開始還沒排，所以是空的
    return tableNext(n, p) # 呼叫 tableNext 遞迴下去排出所有可能

def tableNext(n, p):
    i = len(p)      # i 是下一個排列的位置
    if i == n:          # 全部排好了
        print(p)     # 印出排列
        return      # 返回上層
    for x in [0,1]:     # x 是 0 或 1
        p.append(x)             # 把 x 放進表
        tableNext(n, p)  # 繼續遞迴尋找下一個排列
        p.pop()                 # 把 x 移出表

truthTable(2) # 印出 2 變數的真值表
truthTable(3) # 印出 3 變數的真值表
```

truthtable2.py 計算函數 f 的真值

```
def truthTable(n, f): # 列出 n 變數的所有可能 0,1 排列
```

```
    p = [] # p 代表已經排下去的，一開始還沒排，所以是空的
    return tableNext(n, p, f) # 呼叫 tableNext 遞迴下去排出所有可能

def tableNext(n, p, f):
    i = len(p)      # i 是下一個排列的位置
    if i == n:            # 全部排好了
        print(p, f(p))        # 印出排列
        return      # 返回上層
    for x in [0,1]:     # x 是 0 或 1
        p.append(x)             # 把 x 放進表
        tableNext(n, p, f)      # 繼續遞迴尋找下一個排列
        p.pop()                 # 把 x 移出表

def f(p):   函數 f( )
    x,y,z = p
    return x or (y and z)

print('[x, y, z] f')
truthTable(3, f) # 印出 3 變數的真值表
```