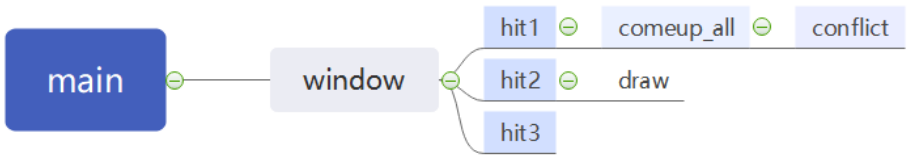
一．编译环境

在 linux，python3.7 下进行编程，运用了 tkinter、numpy、threading、time 模块。

二．代码思路简述

一共 9 个函数，关系如下：



(函数关系图)

1. main 函数

   初始化 n 皇后的 n 值为 0，人为设定了显示窗口的分辨率为 8x8，调用 window()

2. window 函数

   初始化弹窗的静态部分。

   1) 利用 tkinter 绘制窗口，设置长宽和窗口的 title，利用画布设置背景色为白色；
   2) 添加二个文本框、一个标签、三个按钮，每个按钮通过接口分别进入 hit1、hit2、hit3；
   3) 对上述六个零件排版；

3. conflict 函数

   判断加入新的皇后后，棋盘状态是否冲突。

   1) 输入为下一步可能走的位置 pos 和当前棋盘的状态列表 states，其中 pos 是一个数字，表征下一步下在 (len(states), pos) 处；states 为 m x 1 的列表，states[0] 表示第 0 行，第 states[0]列；
   2) 比较 pos 和 states，判断加入 pos 后，是否存在某两个点在同一列或斜率为+-1，返回值为 True 或 False

4. comeup_all 函数

   通过递归来求解。

   1) 在一个 j 属于 0 到 n-1 的 for 循环中，先通过 conflict 函数判断加入 j 后是否冲突，如果不冲突，将该 pos 加入 states 中。
      - 如果此时 states 的长度等于 n，则此时出现一个可行解，把该解加入到最终解集 results 中；pop 掉 states 最后一个 pos，继续进行 for 循环；
      - 如果此时 states 的长度小于 n，进入下一次递归
   2) for 循环结束后，pop 掉 states 的最后一个值，返回上一层

5. hit1 函数

   初始化多个参数，开启多线程，进行求解。

   1) 新建一个小的棋盘画布，覆盖在背景画布上；
   2) 输入 n 后，点击 hit1，根据输入的 n 在棋盘画布上绘制棋盘；保存每个棋格的左上角的坐标；
   3) 利用 threading，开启一个新的线程，运算求解 comeup_all；
   4) 每次输入完新的 n，点击 hit1（确认）按钮后，会更新页面，重新绘图；

6. hit2 函数

1) 内含一个循环计数器 count，用来标记点击 hit 的次数；
2) 每次点击 hit2，通过 while 循环和 try，在 results 寻找对应的 results[count]；
3) 找到 results[count]后，利用 draw 函数绘制在窗口中；
4) 求解的结果会在 text 中显示，可以使用最右侧的 scrollbar 来拖动 text
5) 当逐步求解完所有的解后，再次单击“逐步求解”（hit2）会弹窗提示已显示完所有的解

7. draw 函数
   1) 将上一次的绘制结果用白色覆盖
   2) 将本次找到的结果用黄色绘制
8. Hit3 函数
   根据多线程的结构，运用 messagebox 直接弹窗显示结果。
9. Check_n 函数
   检查是否输入 n 以及检查 n 是否合法（为 int 型的整数）

三．编写代码的收获和遇到的问题

1. 更加了解递归了，最开始一直在尝试用 return，最后发现行不通，思路有问题；不过也算是又去琢磨了一遍递归
2. 了解了把实际任务拆开，再逐步分散求解的处理问题的方法
3. 运用了多次 global，锻炼了逻辑能力
4. 最开始一直在尝试在递归中运用 yield，但调试的时候发现，在递归中，yield 和 next 的组合只能中断和进入一次。查了一些资料，发现也没有很好的组合办法，这次的失败积累了经验，下次写代码时可以避免这种低效的方法；同时也比较详细地了解了 yield。
5. 第一次组合运用了 threading、try 和 sleep，感觉这种组合很厉害，可以很简单地解决一些以前绕了半天才能解决的问题。但是这次没有深入地了解 threading，以后有机会的话要再去好好学学。其中在使用 try 时，一直不知道在 except 或 final 后面写什么，最后查资料发现可以写 pass，学习到了。
6. 初步了解了 tkinter 模块的使用。

四．程序使用说明

1. 首先输入该问题的“n”；
2. 单击“确认”，绘制棋盘；
3. 单击“逐一求解”，可逐步寻找解；
4. 单击“所有解”，若当前时刻已求解完毕，显示所有解；如果没有，显示已求出的解的数量；
5. 在 4 步骤求解完毕后，更改 n，再次执行 2-5，可进行其他问题的求解。