算法流程：

1. **随机生成迷宫大小**
2. **利用Prim算法生成迷宫**（0代表路，1代表墙）【生成迷宫的算法有四种，此处选取Prim】
3. **选取入口和出口**（入口：从第一列正序遍历找到合适位置，入口：从最后一列逆序遍历找到合适位置）
4. **A\*算法**：

说明： OPEN表：存放待拓展的节点

CLOSED：存放拓展过的节点，也就是走过的路

代价F = G + H

G：移动代价(入口到该节点的曼哈顿距离)

H：该节点到出口的移动成本(该节点到出口的曼哈顿距离)）

1. **初始化OPEN表、CLOSED表和代价函数**
2. **选定入口为初始节点，计算初始节点代价H，F= G+H(初始化G为0)**
3. **开始迭代拓展节点，直到拓展节点为出口：**
   1. **从OPEN表中选取代价最小的节点，移除OPEN表，放入CLOSED表**

说明：首先选取F最小的节点，如果F相同，则选取H最小的节点，若都相同，则选取最后生成的OPEN表中最靠后的节点（也就是最后生成的节点，生成顺序：↑，→，↓，←，顺时针）

* 1. **拓展节点**

**寻找该节点x周围不是墙的节点（**顺序：↑，→，↓，←，顺时针）**，将拓展出的节点的父节点置为x,并计算拓展出的节点的F=G+H(H=父节点.H+1)，放入OPEN表**

* 1. **绘制当前图像（OPEN表中节点：蓝色方块；当前扩展节点：绿色方块；CLOSE表中节点：红色方快）**

**5、寻找到出口，迭代找出出口节点的父节点（父节点的父节点。。。。。。。），直到找到入口停止，绘图显示该过程，将该过程中的节点为青色方块**

**（最终，图像显示的蓝色方块为待扩展的节点，红色方块表示拓展过但无效的节点，青色方块表示寻找到的最佳路径）**

**传统方法（深度优先搜索）与A\*算法的唯一区别：不需要计算每个节点的代价函数，直接按照当前节点的拓展顺序（**↑，→，↓，←，**），找能走的路，直接走，如果走到头是死路，则掉头寻找下一个能走的路，直到寻找到出口**

**代码部分文档**

**1.1 代码构成说明：**

**（1）maze\_generator.py：**利用prim算法生成迷宫地图【借鉴他人经验】

调用输入：要生成迷宫的宽和高

调用输出：一个指定大小的迷宫地图（**0为路，1为墙**）、入口位置、出口位置，且迷宫只有一条通路

**（2）maze\_solver.py：**利用A\*算法或深度优先搜索完成迷宫寻路，并通过回调函数显示求解过程。【深度优先借鉴他人经验，A\*算法原创】

调用输入：迷宫地图、入口位置、出口位置，和回调的显示函数

调用输出：通过回调函数进行显示

**（3）maze.py：**主程序，包含控制可视化界面的生成、与用户的操作控制、求解过程的实时显示

**1.2 用到的库**

**Pygame：**用于构建可视化界面

**Threading：**用于多线程编程

**Numpy：**用于涉及到的一些数学计算

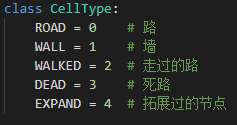
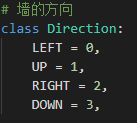
**Random：**用于随机产生迷宫的大小

**Time：**用于控制演示求解过程每一步的时间间隔

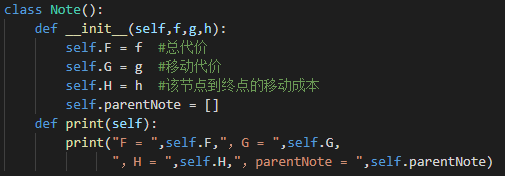
**1.3 maze\_solver.py中A\*算法详细说明**

**（1）使用到的自定义类：**

**基础定义：**



**A\*算法用到的节点定义：**



**坐标定义：**

**X**

**Y**

**（2）A\*算法代码说明**

**①使用到的几个列表：**

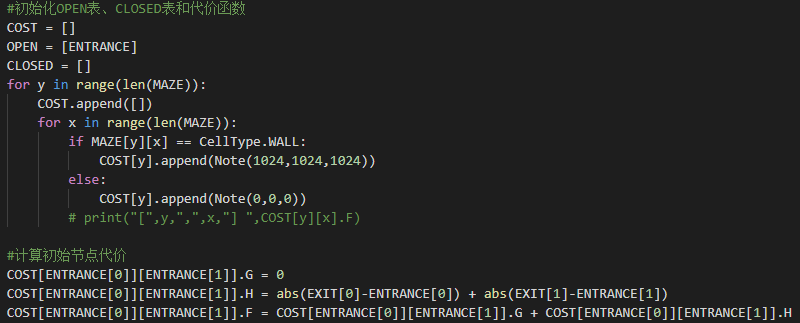
**COST = []：**二维列表，存放每个节点的note对象，包含F、G、H、父节点

**OPEN = []：**存放待扩展的节点坐标，示例：[7,23]、[ 8,23]

**CLOSED = []：**存放已经扩展过的节点，示例：[9,23]、[11,23]、[12,23]

**②算法代码实现流程：**

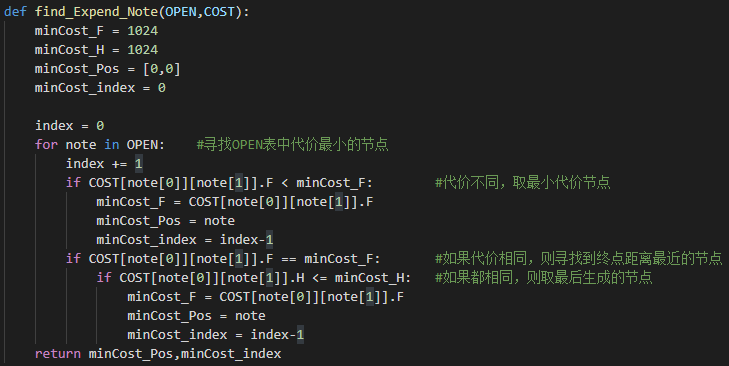
**1.初始化OPEN、CLOSED表，**将入口ENTRANCE直接放入OPEN表里，墙的COST初始化1024表示无限大，路初始化为0。**并计算入口节点的代价函数。**



**2.开始迭代拓展节点，直到拓展节点为出口：**

**a) 从OPEN表中选取代价最小的节点，移除OPEN表，放入CLOSED表**





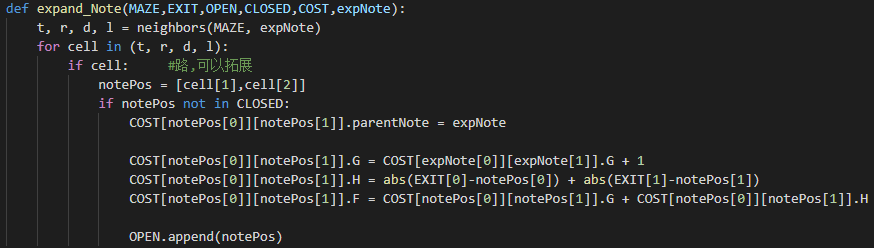
说明：首先选取F最小的节点，如果F相同，则选取H最小的节点，若都相同，则选取最后生成的OPEN表中最靠后的节点（也就是最后生成的节点，生成顺序：↑，→，↓，←，顺时针）

**b) 拓展节点**

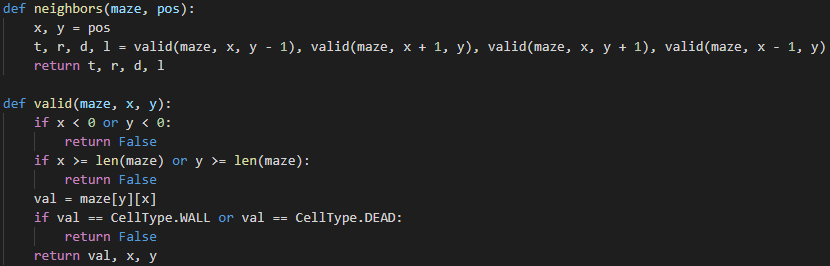
主程序调用：



扩展节点函数：先寻找到要拓展节点周围能扩展的节点（是路且不在CLOSED表中）（顺序：↑，→，↓，←，顺时针），计算拓展出节点的代价函数F=G+H(H=父节点.H+1)，再将其放入CLOSED表中：

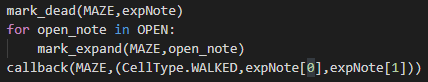


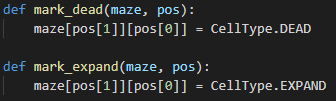
寻找带拓展节点周围可以扩展的节点，按照上右下左（t、r、d、l）顺序搜寻,neighbors()用于拓展，valid()用于检查周围节点是否可以拓展：



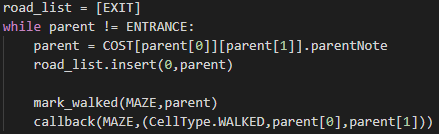
**c) 绘制当前图像**

调用回调函数绘制（回调函数利用迷宫的的标记信息CellType进行绘制），（OPEN表中节点：蓝色方块；当前扩展节点：绿色方块；CLOSE表中节点：红色方快）





**3、寻找到出口，迭代找出出口节点的父节点，直到找到入口停止，绘图显示该过程。**



Road\_list用于存放找出的路径信息，示例如下：

