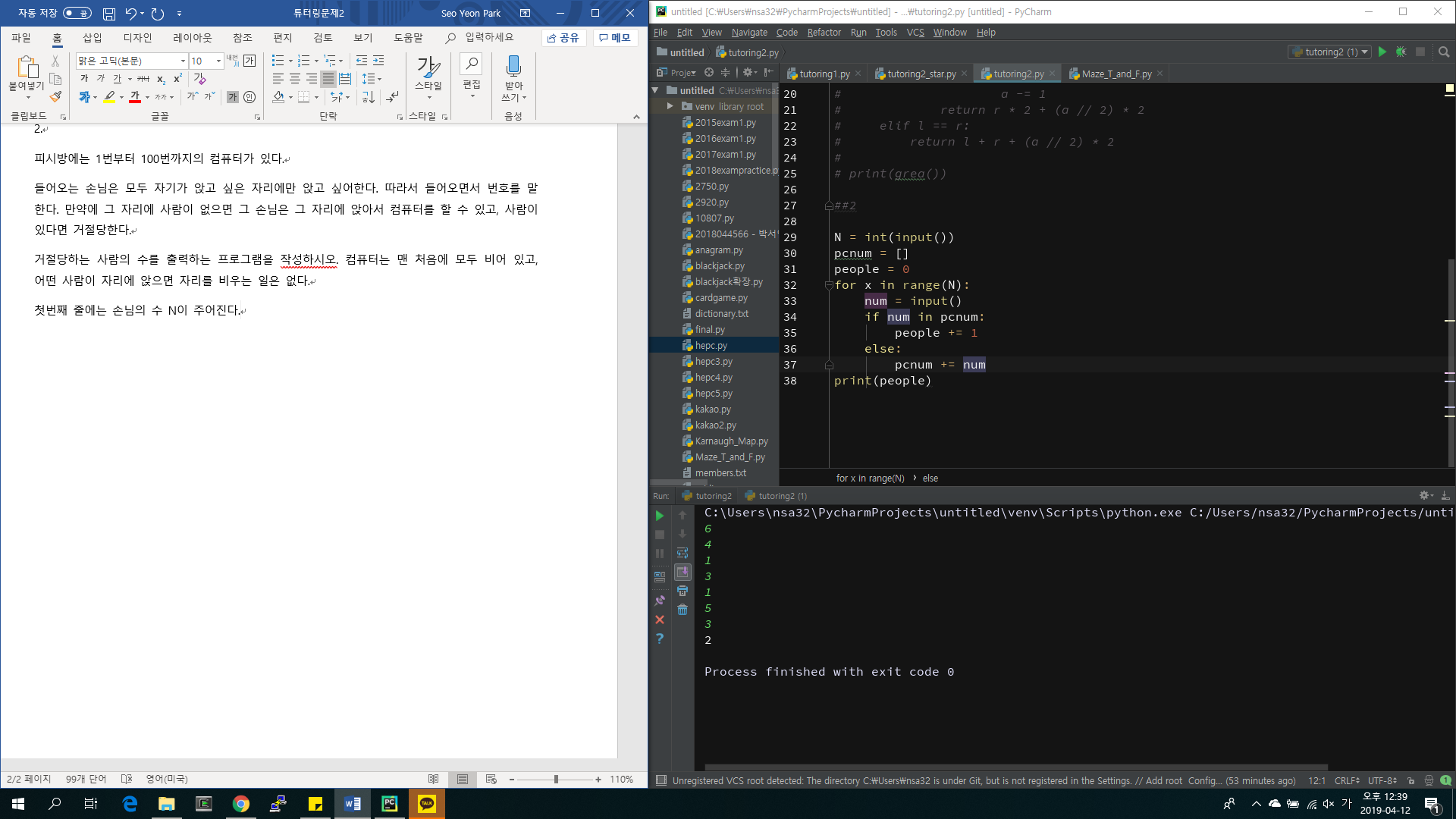
피시방에는 1번부터 100번까지의 컴퓨터가 있다.

들어오는 손님은 모두 자기가 앉고 싶은 자리에만 앉고 싶어한다. 따라서 들어오면서 번호를 말한다. 만약에 그 자리에 사람이 없으면 그 손님은 그 자리에 앉아서 컴퓨터를 할 수 있고, 사람이 있다면 거절당한다.

거절당하는 사람의 수를 출력하는 프로그램을 작성하시오. 컴퓨터는 맨 처음에 모두 비어 있고, 어떤 사람이 자리에 앉으면 자리를 비우는 일은 없다.

첫번째 줄에는 손님의 수 N이 주어진다.



미로 찾기

다음 그림과 같은 미로가 있다. 한번도 가보지 않은 길은 0으로 벽은 1로 표시가 되어있다. 출구는 오른쪽 맨 아래칸이라 할 때, 길을 찾다가 막혀 있는 길이라면 3으로 표시를 하고, 막혀있지 않은 출구까지의 길이라면(즉, 미로 찾기의 정답 경로이면) 2로 표시를 한다고 할 때, 주어진 미로 maze1, maze2, maze3, maze4에 출구까지의 길이 있는지 확인하는 함수 findPath(maze,x,y)의 빈 부분을 작성하시오.(주어진 미로에 출구까지의 길이 있다면 findPath는 True를 리턴하고, 그렇지 않으면 False를 반환한다.)

0 : UNVISITED\_WAY

1 : WALL

2 : POSSIBLE\_WAY

3 : BLOCKED

**START**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 3 | 2 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

**FINISH**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 3 | 3 | 3 | 3 | 3 | 1 |
| 2 | 1 | 1 | 3 | 1 | 1 | 3 | 1 |
| 2 | 3 | 3 | 1 | 3 | 3 | 3 | 1 |
| 2 | 1 | 3 | 3 | 1 | 1 | 3 | 3 |
| 2 | 1 | 1 | 1 | 3 | 3 | 1 | 1 |
| 2 | 1 | 2 | 2 | 2 | 1 | 3 | 1 |
| 2 | 2 | 2 | 1 | 2 | 2 | 2 | 1 |
| 3 | 1 | 1 | 1 | 3 | 1 | 2 | 2 |

def findPath(maze,x,y):  
 if(x>7 or y>7 or x<0 or y<0):  
 return False  
 elif(maze[x][y] != 0): *# 1, 2, 3 NOT UNVISITED WAY* return False  
 elif(x == 7 and y == 7):  
 maze[x][y] = 2  
 return True  
 maze[x][y] = 2  
 if( **//fill here!** ):  
 return True  
 else:  
 maze[x][y] = 3  
 return False

maze1 = [[0,0,0,0,0,0,0,1],  
 [0,1,1,0,1,1,0,1],  
 [0,0,0,1,0,0,0,1],  
 [0,1,0,0,1,1,0,0],  
 [0,1,1,1,0,0,1,1],  
 [0,1,0,0,0,1,0,1],  
 [0,0,0,1,0,0,0,1],  
 [0,1,1,1,0,1,0,0]]  
print(findPath(maze1,0,0))