왕실의 나이트

#나이트 이동

dx=[-2, -2,+2,+2,+1,-1,+1,-1 ]

dy=[1, -1, 1, -1,+2,+2,-2,-2]

전 문제처럼 위치마다 인덱스를 붙여놓은 것이 아니라(R, L, U, D처럼)

위와 같이 푸는 것 보단

step=[(-2,-1),(-2,1),(2,-1),(2,1),(-1,2),(1,2),(-1,-2),(1,-2)]

이렇게 해서 i[0], i[1]로 접근하는 것이 나음

두 가지 모두 자주 사용되므로 참고하기.

-아스키코드로 변환시 ord()를 사용해주어야 한다.

바로 int(a)하면 오류남..

-나의 풀이(정답)

id=input()

x=int(id[1])

y=int(ord(id[0]))-97+1

cnt=0

step=[(-2,-1),(-2,1),(2,-1),(2,1),(-1,2),(1,2),(-1,-2),(1,-2)]

for i in step:

    cx=x+i[0]

    cy=y+i[1]

    if cx<1 or cy<1 or cx>8 or cy>8:

        continue

    cnt+=1

print(cnt)

---

시뮬레이션 문제는 삼전 코테에 자주 출제됨.

방향 설정 문제는 dx,dy 별도 리스트를 만드는 것이 효과적

# N, M을 공백을 기준으로 구분하여 입력받기

n, m = map(int, input().split())

# 방문한 위치를 저장하기 위한 맵, 0으로 초기화

d = [[0] \* m for \_ in range(n)]

# 현재 캐릭터의 X 좌표, Y 좌표, 방향을 입력받기

x, y, direction = map(int, input().split())

d[x][y] = 1 # 현재 좌표 방문 처리

# 전체 맵 정보를 입력받기

array = []

for i in range(n):

    array.append(list(map(int, input().split())))

# 북, 동, 남, 서 방향 정의

dx = [-1, 0, 1, 0]

dy = [0, 1, 0, -1]

# 왼쪽으로 회전

def turn\_left():

    global direction

    direction -= 1

    if direction == -1:

        direction = 3

**하지만, for 문을 리스트에 내포함으로써 좀 더 직관적인 프로그램을 만들 수 있는데요.**

**다음은 리스트 내포 for 문의 일반적인 문법입니다.**

**if 문은 생략이 가능하며, for 문을 여러 개 사용할 수도 있습니다.**

[ 표현식 for 항목 in 리스트 or 튜플 if 조건문 ]

**다음은 리스트 각 항목에 3을 곱한 결과를 다시 원래 리스트에 대입하는 예제 코드입니다.**

list = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]

# 리스트 안에서 for 문 사용하기 1

list = [ num \* 3 for num in list ]

print(list)



**if 문 역시 리스트 안에서 for 문과 연계하여 사용할 수 있는데**

**다음은 리스트 각 항목 중 홀수만 따로 뽑아서 3을 곱한 결과를 다시 원래 리스트에 대입하는 예제 코드입니다.**

list = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]

# 리스트 안에서 for 문 사용하기 2: if 문 사용

list = [ num \* 3 for num in list if num % 2 == 1 ]

print(list)

# 시뮬레이션 시작

count = 1

turn\_time = 0

while True:

    # 왼쪽으로 회전

    turn\_left()

    nx = x + dx[direction]

    ny = y + dy[direction]

    # 회전한 이후 정면에 가보지 않은 칸이 존재하는 경우 이동

    if d[nx][ny] == 0 and array[nx][ny] == 0:

        d[nx][ny] = 1

        x = nx

        y = ny

        count += 1

        turn\_time = 0

        continue

    # 회전한 이후 정면에 가보지 않은 칸이 없거나 바다인 경우

    else:

        turn\_time += 1

    # 네 방향 모두 갈 수 없는 경우

    if turn\_time == 4:

        nx = x - dx[direction]

        ny = y - dy[direction]

        # 뒤로 갈 수 있다면 이동하기

        if array[nx][ny] == 0:

            x = nx

            y = ny

        # 뒤가 바다로 막혀있는 경우

        else:

            break

        turn\_time = 0

# 정답 출력

print(count)