

## 슬라이딩 퍼즐 Sliding Puzzle


슬라이딩 퍼즐은 정사각형 보드에 나열된 납작한 조각판을 좌우상하로만 움직여 정해진 모양으로 맞추는 퍼즐보드 게임이다. 이 퍼즐게임 프로그램을 공부하자.

### 1단계 : 코드 리뷰

강의비디오 5-4 슬라이딩퍼즐.mp4, 5-5 슬라이딩퍼즐 코딩라이브.mp4를 찾아서 아래 설명을 보면서 시청한다. 이 강의비디오는 4x4 크기의 퍼즐보드에서 1~15의 수로 구성된 조각판을 움직이는 슬라이딩 게임이며, 아래에 기술한 요구사항 및 구현전략에 따라서 코드를 작성하는 과정을 보여 준다.

- 초기에 슬라이딩 퍼즐보드는 아래의 왼쪽 그림과 같이 첫 칸이 비어있고 나머지 칸은 15부터 1까지 내림차순으로 채워져 있다. 빈칸 옆에 있는 번호는 밀어서 빈칸으로 움직일 수 있다. 즉, 초기 퍼즐보드에서 15는 왼쪽으로 12는 위쪽으로 움직일 수 있다. 슬라이드 퍼즐 게임은 아래의 왼쪽과 같은 퍼즐보드(초기보드)로 시작하여 오른쪽과 같은 퍼즐보드(해답보드)를 만들면 끝난다.

	0	1	2	3
0		15	14	13
1	12	11	10	9
2	8	7	6	5
3	4	3	2	1



	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	

- 슬라이드 퍼즐 게임의 진행 순서는 다음과 같다.
  - (컴퓨터) 퍼즐 보드를 보여준다.
  - (컴퓨터) 퍼즐 보드가 위 오른쪽에 주어진 모양대로 1부터 15까지 오름차순으로 채워지면 게임을 끝내고, 그렇지 않으면 C로 간다.
  - (플레이어) 빈칸과 이웃한 번호 중에서 움직이고 싶은 번호를 입력한다.
  - (컴퓨터) 플레이어에게서 받은 번호를 빈칸으로 밀어 움직이고 A로 돌아간다.
- 퍼즐보드는 아래 그림과 같이 2차원 행렬 테이블로 볼 수 있으며, 각 행과 열의 위치는 튜플좌표로 표현할 수 있다.

	0	1	2	3
0	(0,0)	(0,1)	(0,2)	(0,3)
1	(1,0)	(1,1)	(1,2)	(1,3)
2	(2,0)	(2,1)	(2,2)	(2,3)
3	(3,0)	(3,1)	(3,2)	(3,3)

- 그러면 퍼즐보드는 프로그램에서 어떻게 표현할까? 다음의 초기보드를 예로 들어 표현해보자.

	0	1	2	3
0		15	14	13
1	12	11	10	9
2	8	7	6	5
3	4	3	2	1

행마다 하나의 리스트를 만들고, 그 리스트의 리스트로 다음과 같이 표현할 수 있다. 빈칸은 0으로 표현한다.

```
board = [[0, 15, 14, 13], [12, 11, 10, 9], [8, 7, 6, 5], [4, 3, 2, 1]]
```

셋째줄(2번째)인 [8, 7, 6, 5]는 다음과 같이 꺼낼 수 있다.

```
board[2]
```

셋째줄(2번째)의 둘째값(1번째)인 7은 다음과 같이 꺼낼 수 있다.

```
board[2][1]
```

- 이 게임을 Python 함수로 다음과 같이 구현할 수 있다.

```

1 def sliding_puzzle():
2     board = create_init_board()
3     goal = create_goal_board()
4     empty = (0, 0)
5     while True:
6         print_board(board)
7         if board == goal:
8             print("Congratulations!")
9             break
10        num = get_number()
11        if num == 0:
12            break
13        pos = find_position(num, board)
14        (empty, board) = move(pos, empty, board)
15    print("Please come again.")
```

- 굵게 표시된 함수는 모두 자가제작 함수로서 각 함수의 임무는 아래에 각각 간단히 설명한다. 비디오를 시청하며 주어진 코드를 리뷰하면서 코드를 완전히 이해하자.

#### **create\_init\_board()**

- 초기보드를 만들어 내주는 함수

#### **create\_goal\_board(board)**

- 해답보드를 만들어 내주는 함수

**print\_board(board)**

- 중첩리스트로 표현된 퍼즐보드 board를 인수로 받아서 다음과 같이 실행창에 프린트하는 함수

```
>>> board = create_init_board()
>>> print_board(board)
15 14 13 12
11 10  9  8
 7  6  5  4
 3  2  1
```

**get\_number()**

- 실행창에서 메시지를 보여주며 사용자에게서 0부터 15사이의 입력을 받아서 정수로 바꾸어 내주는 함수
- 0부터 15사이의 입력이 아닌 경우에는 정상 입력을 받을 때까지 반복해서 입력 확인

```
>>> get_number()
Type the number you want to move (Type 0 to quit): seven
Type the number you want to move (Type 0 to quit): 16
Type the number you want to move (Type 0 to quit): 5
5
```

**find\_position(num, board)**

- 1부터 15사이의 정수 num과 퍼즐 보드 board를 인수로 받아 num의 위치 좌표를 행번호와 열번호의 튜플로 내주는 함수

```
>>> board = create_init_board()
>>> print_board(board)
15 14 13 12
11 10  9  8
 7  6  5  4
 3  2  1
>>> find_position(9, board)
(1, 2)
>>> find_position(3, board)
(3, 0)
>>> find_position(1, board)
(3, 2)
```

**move(pos, empty, board)**

- 이동할 번호의 위치 좌표 pos, 빈칸의 좌표 empty와 퍼즐 보드 board를 인수로 받아, pos가 empty와 이웃해 있는지 확인하고, board와 empty의 내용을 서로 바꾸고 바뀐 empty와 board의 튜플 쌍을 내준다. pos가 empty와 이웃해 있지 않으면 "Can't move! Try again!" 라는 메시지를 출력하고 바뀌지 않은 empty와 board의 튜플 쌍을 내준다. 실행 예를 보자.

```
>>> print_board(board)
15 14 13 12
```

```

11 10 9 8
7 6 5 4
3 2 1
>>> opened = (3, 3)
>>> move((2, 2), opened, board)
Can't move! Try again.
((3, 3), [[15, 14, 13, 12], [11, 10, 9, 8], [7, 6, 5, 4], [3, 2, 1, 0]])
>>> move((2, 3), opened, board)
((2, 3), [[15, 14, 13, 12], [11, 10, 9, 8], [7, 6, 5, 0], [3, 2, 1, 4]])
>>> print_board(board)
15 14 13 12
11 10 9 8
7 6 5
3 2 1 4

```

주어진 slidingpuzzle.py 파일에 완성된 4x4 슬라이딩 퍼즐 게임이 있다. 이 코드를 실행해보고, 코드 리뷰를 하여 코드를 완전히 이해하자.

## 2단계 : 코드 개선

1단계 코드 리뷰 완료 후, 요구사항 변경에 맞추어 코드를 개선하여 완성한 코드를 제출한다.

### 수정 요구사항

1. 사용자가 게임보드의 크기를 지정할 수 있도록 한다. 즉, 2x2, 3x3, 4x4, 5x5, ... 와 같이 제한없이 선택하고 선택한 크기의 보드에서 퍼즐게임을 할 수 있어야 한다. 게임을 시작할 때 명령창에서 보드 크기를 시스템 인수로 지정한다. 보드의 크기를 5x5로 지정하는 경우, 명령은 다음과 같다.

```
$ python3 slidingpuzzle.py 5
```

2. 지금은 초기 퍼즐보드가 고정식이다. 사용자가 다양하게 도전을 해볼 수 있도록 초기 퍼즐보드를 무작위로 섞어서 내주도록 수정한다. create\_init\_board 함수를 무작위로 수가 배열된 보드를 내주도록 수정해야 한다.

### 시스템 인수 지정 방법

명령창에서 인수를 프로그램에 전달하기 위해서는 표준 라이브러리의 sys 모듈의 argv를 쓴다. 프로그램의 맨 아래에 다음과 같이 기술하고,

```

if __name__ == "__main__":
    import sys
    size = int(sys.argv[1])

```

다음과 같이 명령창에서 다음과 같은 명령을 내리면,

```
$ python3 slidingpuzzle.py 5
```

sys.argv[0]의 값은 "slidingpuzzle.py"로 지정되고, sys.argv[1]의 값은 "5"로 지정되어 프로그램에서 사용할 수 있다. 인수가 더 있으면 argv 리스트에 추가로 나열되어 sys.argv[2], sys.argv[3] 등

에 문자열로 지정된다. 위의 사례의 경우, `sys.argv[1]`에 지정된 문자열 "5"를 정수로 타입변환하여 `size` 변수로 지정하여 사용하도록 하였다. 이 퍼즐게임의 보드 크기는 사용자가 지정한 시스템 인수 값에 따라 변해야 하므로, `slidingpuzzle` 함수를 호출하면서 `size` 값을 다음과 같이 인수로 전달해주어야 할 것이다.

```
slidingpuzzle(size)
```

명령창에서 `sys.argv[1]`로 전달받은 인수도 외부에서 들어오는 값이니 보안을 위하여 입력확인을 해야 한다. 통과조건은 숫자문자로만 구성되고 정수로 2 이상이다. 따라서 입력확인까지 완성한 코드는 다음과 같다.

```
if __name__ == "__main__":
    import sys
    size = sys.argv[1]
    if size.isdigit() and int(size) > 1:
        sliding_puzzle(int(size))
    else:
        print("Not a proper system argument.")
```

이제 코드를 주어진 코드의 맨 아래줄의 다음 코드를 대치하고, 코딩을 시작하자.

```
sliding_puzzle()
```

## 코딩 가이드

4로 고정되어 있었던 보드의 크기를 이젠 `sliding_puzzle` 함수에 `size` 인수로 아래와 같이 전달한다. 이 보드의 크기는 아래 코드와 같이 필요한 적재적소에 제공해주어야 한다. 이 함수에서 변경할 부분은 짚게 표시한 부분이다. 이제 맞게 해당 함수를 수정한다. 줄 4도 수정되었다. `create_init_board` 함수가 이제는 무작위 보드를 내주도록 수정해야 하므로 빈칸의 위치도 이젠 고정되어 있지 않아서 0이 있는 위치를 찾아주어야 하기 때문이다.

```
1 def sliding_puzzle(size):
2     board = create_init_board(size)
3     goal = create_goal_board(size)
4     empty = find_position(0, board)
5     while True:
6         print_board(board)
7         if board == goal:
8             print("Congratulations!")
9             break
10        num = get_number(size)
11        if num == 0:
12            break
13        pos = find_position(num, board)
14        (empty, board) = move(pos, empty, board)
15        print("Please come again.")
```