[COM1002]
프로그래밍1

퍼즐게임 스도쿠

#08. 프로젝트 기반 학습 I

김현하

한양대학교 ERICA 소프트웨어학부

2021.11.16.
2021년도 2학기

스도쿠

스도쿠

- 스도쿠數獨
  - 가로 9칸, 세로 9칸의 9x9 격자 보드에 1부터 9까지의 숫자를 일정한 규칙에 맞게 채워 넣는 퍼즐
  - 數셀 수獨홀로 독: 숫자는 한 번씩만 쓸 수 있다
  - 일본에서 만든 퍼즐 책이 널리 퍼지면서 2005년부터 전 세계적으로 유행

스도쿠

- 스도쿠의 규칙
  - 9개의 세로 줄과 9개의 가로 줄에 1에서 9까지의 숫자를 하나씩만 넣어서 겹치지 않아야 한다
  - 3x3의 단위로 9등분한 격자 보드에도 1에서 9까지의 숫자를 하나씩만 넣어서 겹치지 않아야 한다

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

스도쿠 퍼즐의 예

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

스도쿠 퍼즐의 규칙

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

정답

출처 : *https://ko.wikipedia.org/wiki/스도쿠*

목차

- 중첩 루프
  - 버블정렬
  - 기수정렬
- 퍼즐게임 스도쿠

중첩 루프

중첩 루프

- 중첩 루프nested loop
  - 루프의 코드 블록 안에 루프를 포함하는 구조
  - 행렬과 같은 2차원 이상의 데이터 구조 처리 등 다양한 문제를 푸는데 효과적으로 활용

```
1  couples = []
2  for c in ["a", "b"]:
3      for n in range(3):
4          couple = (c, n)
5          couples.append(couple)
6  print(couples)
```

아스키 아트

아스키 아트

- 아스키 아트ASCII art : 아스키 코드 문자 또는 기호를 적절히 배치하여 그린 그림

가로

```
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
```

```
1  def digit_art_horizontal(n):
2      for i in range(n):
3          for j in range(n):
4              print(j + 1, end=' ')
5          print()
```

세로

```
1 1 1 1 1 1 1
2 2 2 2 2 2 2
3 3 3 3 3 3 3
4 4 4 4 4 4 4
5 5 5 5 5 5 5
6 6 6 6 6 6 6
7 7 7 7 7 7 7
```

```
1  def digit_art_vertical(n):
2      for i in range(n):
3          for j in range(n):
4              print(i + 1, end=' ')
5          print()
```

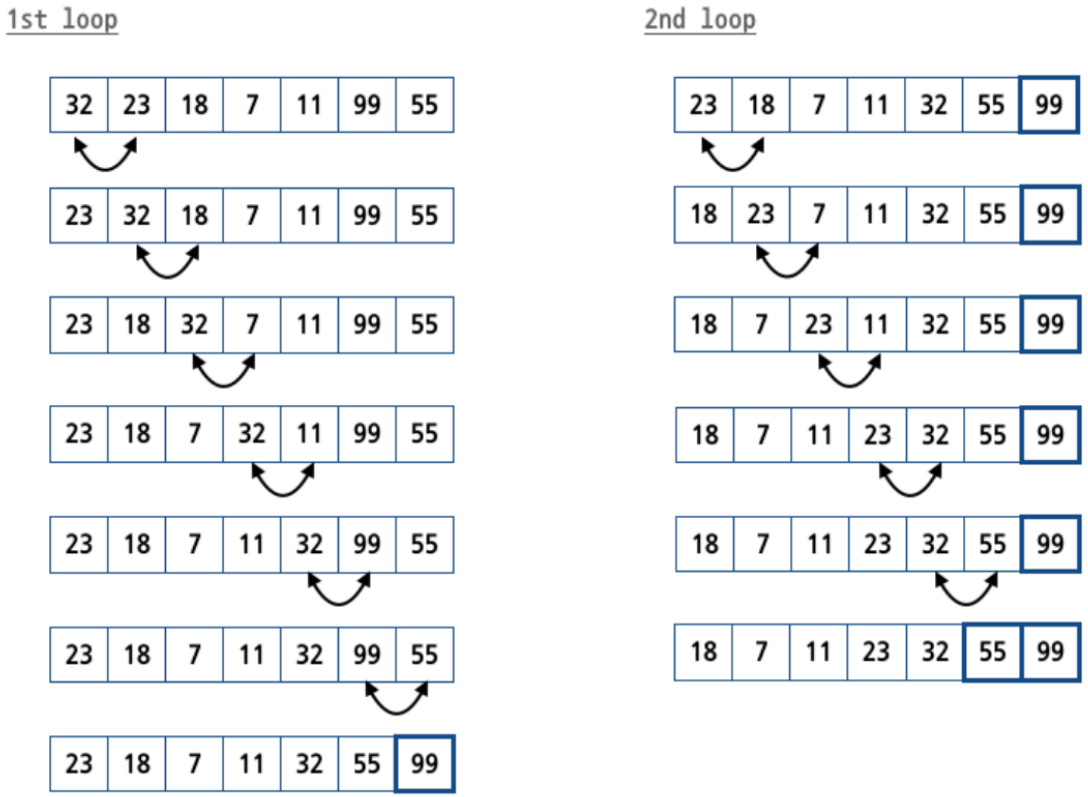
아스키 아트

```
1  def digit_art_horizontal_alternate(n):
2      for i in range(n):
3          for j in range(n):
4              if i % 2 == 0:
5                  print(j + 1, end=' ')
6              else:
7                  print(n - j, end=' ')
8          print()
```

버블정렬

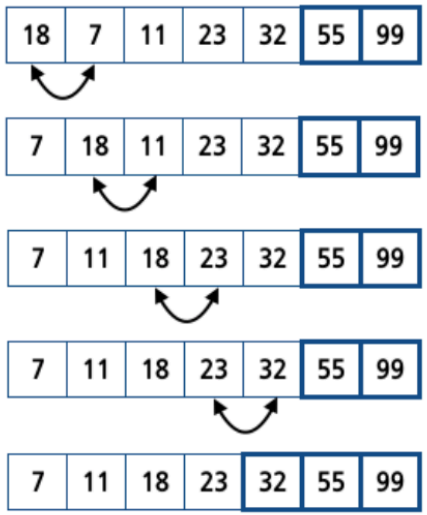
버블정렬

- 버블정렬bubble sort
  - 리스트에서 인접한 두 수를 비교하여 순서가 뒤바뀐 경우 이를 교환하여 정렬하는 방법
  - 5장에서 배운 정렬 알고리즘과 달리 추가 공간을 사용하지 않는 제자리정렬in-place sort

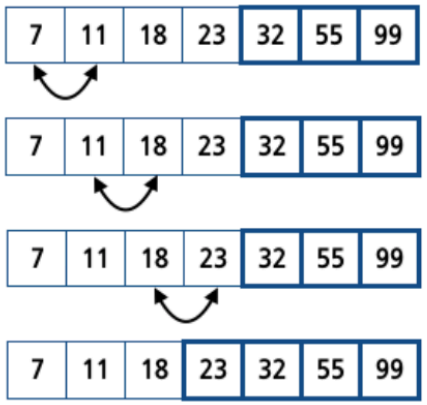


버블정렬

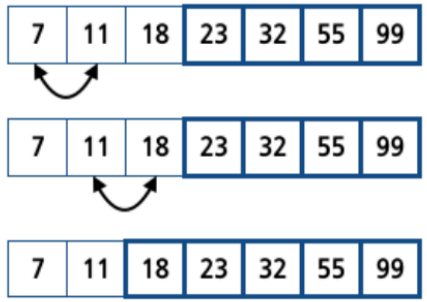
3rd loop



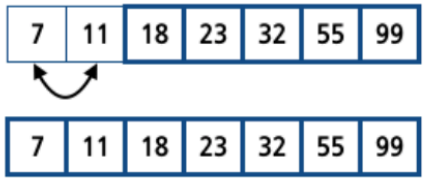
4th loop



5th loop



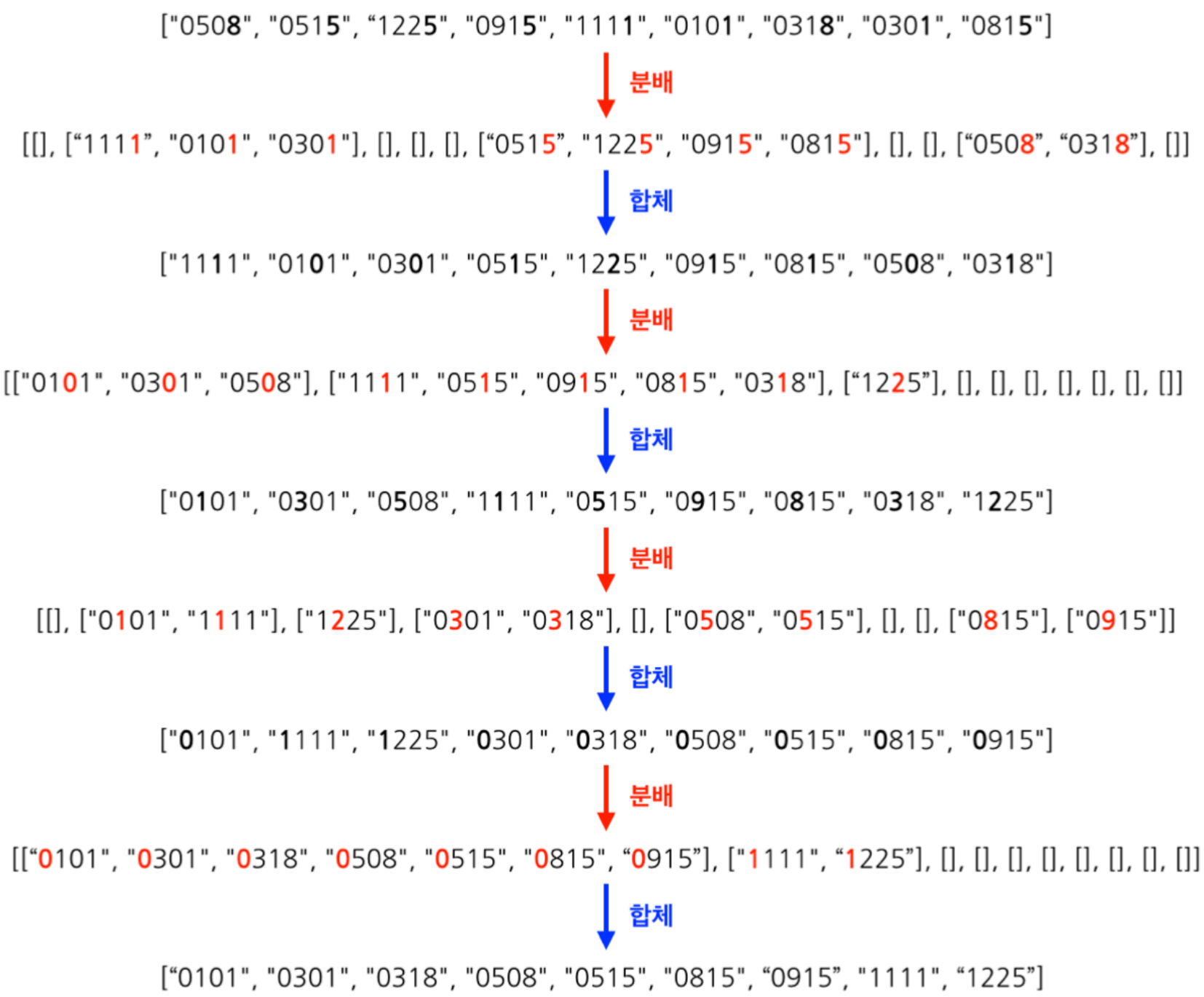
6th loop



기수정렬

기수정렬

- 기수정렬radix sort
  - 길이가 모두 같은 숫자 문자열(주민등록번호, 휴대전화번호, 학번 등)을 정렬할 때, 숫자 크기의 비교없이 효율적으로 정렬하는 방법
  - 낮은 자리 수부터 시작하여 모든 자리수를 차례로 숫자 별로 분배하면 저절로 정렬이 됨


["0508", "0515", "1225", "0915", "1111", "0101", "0318", "0301", "0815"]
분배
[[], ["1111", "0101", "0301"], [], [], [], ["0515", "1225", "0915", "0815"], [], [], ["0508", "0318"], []]
합체
["1111", "0101", "0301", "0515", "1225", "0915", "0815", "0508", "0318"]
분배
[["0101", "0301", "0508"], ["1111", "0515", "0915", "0815", "0318"], ["1225"], [], [], [], [], [], [], []]
합체
["0101", "0301", "0508", "1111", "0515", "0915", "0815", "0318", "1225"]
분배
[[], ["0101", "1111"], ["1225"], ["0301", "0318"], [], ["0508", "0515"], [], [], ["0815"], ["0915"]]
합체
["0101", "1111", "1225", "0301", "0318", "0508", "0515", "0815", "0915"]
분배
[["0101", "0301", "0318", "0508", "0515", "0815", "0915"], ["1111", "1225"], [], [], [], [], [], [], [], []]
합체
["0101", "0301", "0318", "0508", "0515", "0815", "0915", "1111", "1225"]

퍼즐게임 스도쿠

4x4 미니 스도쿠

- 요구사항
  - 가로 4칸, 세로 4칸의 정사각형 보드에 1부터 4까지 숫자를 넣는다.
  - 전체 보드의 가로/세로줄 모두 1에서 4까지의 숫자를 하나씩만 넣는다.
  - 2x2으로 4등분한 조각 보드에도 1에서 4까지의 숫자를 하나씩만 넣는다.
  - 빈칸이 많을 수록 퍼즐 풀이가 어려워 지므로, 난이도를 선택할 수 있게 한다.
    - 초급 : 빈칸 6개 / 중급 : 빈칸 8개 / 고급 : 빈칸 10개

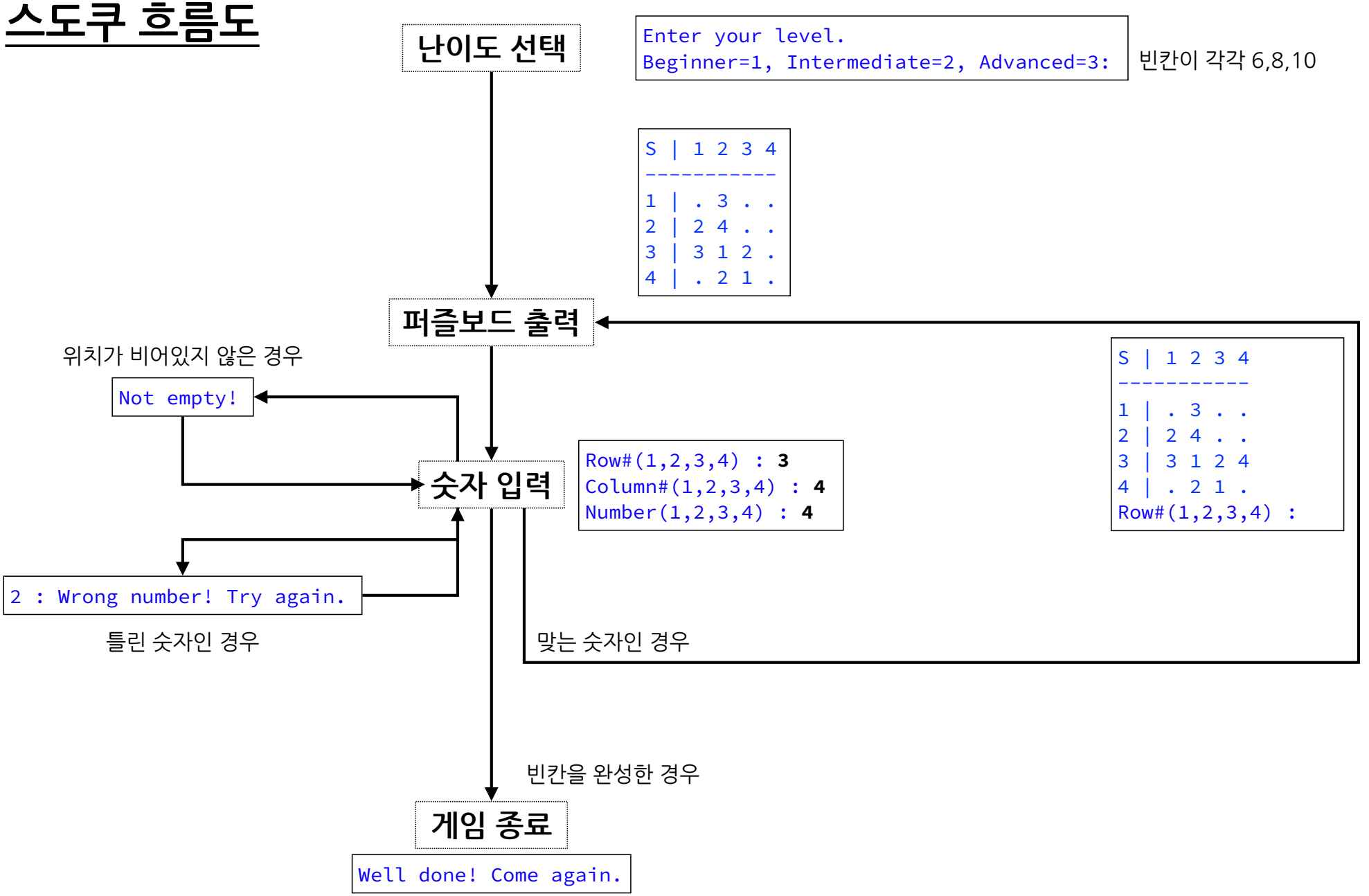
3			2
	4	1	
	3	2	
4			1

퍼즐보드

3	1	4	2
2	4	1	3
1	3	2	4
4	2	3	1

정답보드

스도쿠 흐름도


난이도 선택
Enter your level.
Beginner=1, Intermediate=2, Advanced=3:
빈칸이 각각 6,8,10
S | 1 2 3 4
----
1 | . 3 . .
2 | 2 4 . .
3 | 3 1 2 .
4 | . 2 1 .
퍼즐보드 출력
위치가 비어있지 않은 경우
Not empty!
숫자 입력
Row#(1,2,3,4) : 3
Column#(1,2,3,4) : 4
Number(1,2,3,4) : 4
S | 1 2 3 4
----
1 | . 3 . .
2 | 2 4 . .
3 | 3 1 2 4
4 | . 2 1 .
Row#(1,2,3,4) :
2 : Wrong number! Try again.
틀린 숫자인 경우
맞는 숫자인 경우
빈칸을 완성한 경우
게임 종료
Well done! Come again.

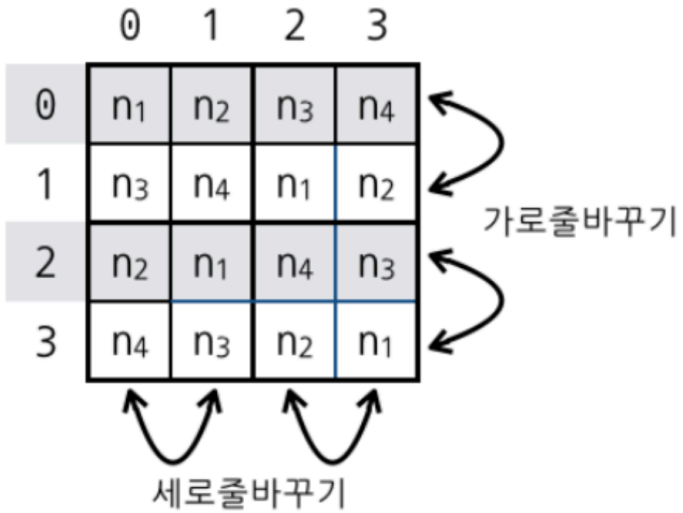
스도쿠 보드의 표현

	0	1	2	3
0	1	2	3	4
1	3	4	1	2
2	2	1	4	3
3	4	3	2	1

```
board = [
   [1, 2, 3, 4],
   [3, 4, 1, 2],
   [2, 1, 4, 3],
   [4, 3, 2, 1]
]
```

- [Python 인터프리터]
  - `row0 = [1,2,3,4]`
  - `row1 = row0[2:4] + row0[0:2]`
  - `row2 = [row0[1], row0[0], row0[3], row0[2]]`
  - `row3 = row2[2:4] + row2[0:2]`
  - `board = [row0, row1, row2, row3]`

스도쿠 정답 보드 만들기



- 정답 보드 줄 교환
  - 가로줄 0 ⟷ 가로줄 1
  - 가로줄 2 ⟷ 가로줄 3
  - 세로줄 0 ⟷ 세로줄 1
  - 세로줄 2 ⟷ 세로줄 3

- 정답 보드의 가지 수
  - $n_1\ n_2\ n_3\ n_4$: $4! = 24$
  - 줄교환 : $2^4 = 16$
  - 24 * 16 = 384 가지

- 정답 보드의 생성
  - [1, 2, 3, 4] 를 무작위로 섞어서 `row0` 생성
  - 줄 교환 여부를 무작위로 적용

스도쿠 정답 보드 만들기

```
1  def initialize_board_4x4():
2      row0 = [1,2,3,4]
3      random.shuffle(row0) # 리스트의 순서를 무작위로 섞음
4      row1 = row0[2:4] + row0[0:2]
5      row2 = [row0[1], row0[0], row0[3], row0[2]]
6      row3 = row2[2:4] + row2[0:2]
7      return [row0, row1, row2, row3]
```

```
1  def shuffle_ribbons(board):
2      top = board[:2]
3      bottom = board[2:]
4      random.shuffle(top)
5      random.shuffle(bottom)
6      return top + bottom
```

스도쿠 정답 보드 만들기

```
1  def initialize_board_4x4():
2      row0 = [1,2,3,4]
3      random.shuffle(row0) # 리스트의 순서를 무작위로 섞음
4      row1 = row0[2:4] + row0[0:2]
5      row2 = [row0[1], row0[0], row0[3], row0[2]]
6      row3 = row2[2:4] + row2[0:2]
7      return [row0, row1, row2, row3]
```

```
1  def shuffle_ribbons(board):
2      top = board[:2]
3      bottom = board[2:]
4      random.shuffle(top)
5      random.shuffle(bottom)
6      return top + bottom
```

알고리즘

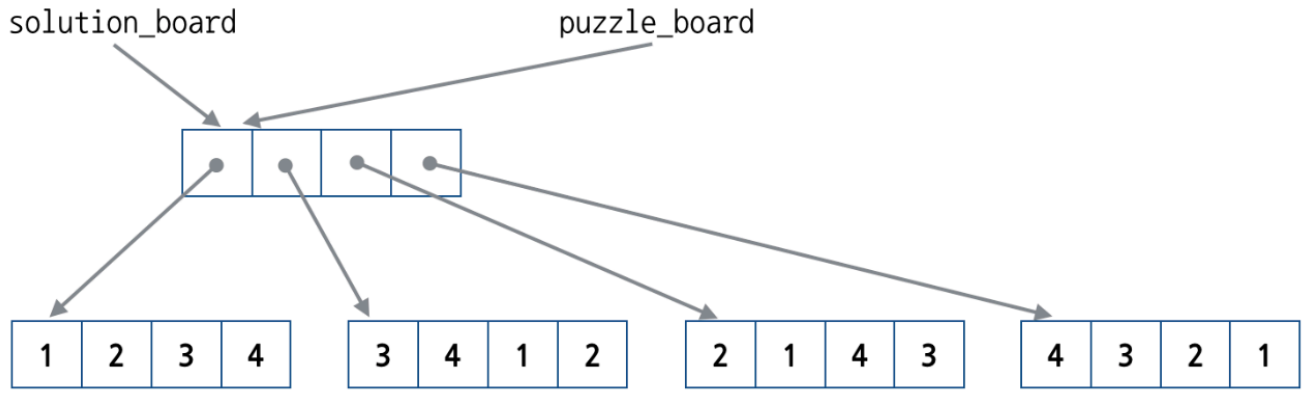
- 무작위로 스도쿠 정답보드 solution_board를 만든다.
- solution_board를 복제하여 puzzle_board를 하나 만든다.
- 사용자에게 난이도를 선택하게 하여 빈칸의 개수 no_of_holes를 정한다.
- puzzle_board에 no_of_holes만큼 무작위로 선택하여 0으로 채운다.
- puzzle_board를 실행창에 정한 형식대로 보여준다. 실행창에는 빈칸을 (0이 아닌) 점으로 표시한다.
- 다음 절차를 no_of_holes가 0이 될 때까지 반복한다.
    - 숫자를 채울 빈칸의 가로줄번호 i, 세로줄번호 j를 차례로 입력받는다.
    - (i,j) 위치에 있는 숫자가 0이 아니면 빈칸이 아니므로 재입력받는다.
    - 빈칸이면, 숫자(1,2,3,4) n을 입력받는다.
    - n이 solution_board[i][j]와 같으면, puzzle_board[i][j]에 그 숫자를 채우고, 갱신한 puzzle_board를 보여준다.
    - 이 숫자가 solution_board[i][j]와 다르면, 줄 번호부터 모두 다시 재입력 받는다.

메인 프로시저 구현

```
1   def sudoku_mini():
2       solution_board = create_solution_board_4x4()
3       puzzle_board = copy_board(solution_board)
4       no_of_holes = get_level()
5       puzzle_board = make_holes(puzzle_board, no_of_holes)
6       show_board(puzzle_board)
7       while no_of_holes > 0:
8           i = get_integer("Row#(1,2,3,4): ", 1, 4) - 1
9           j = get_integer("Column#(1,2,3,4): ", 1, 4) - 1
10          if puzzle_board[i][j] != 0:
11              print("Not empty!")
12              continue
13          n = get_integer("Number(1,2,3,4): ", 1, 4)
14          if n == solution_board[i][j]:
15              puzzle_board[i][j] = n
16              show_board(puzzle_board)
17              no_of_holes -= 1
18          else:
19              print(n, ": Wrong number! Try again.")
20      print("Well done! Come again.")
```

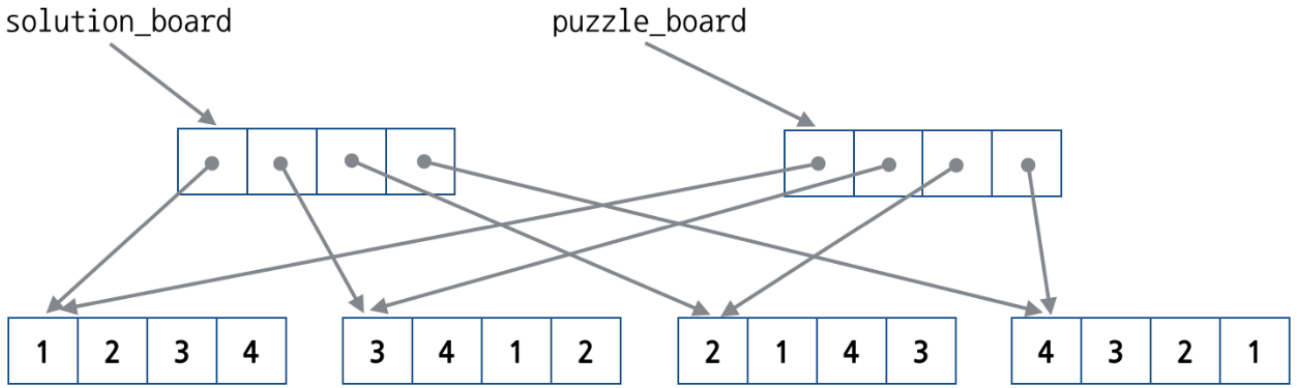
퍼즐보드 복제하기

```
puzzle_board = solution_board
```



퍼즐보드 복제하기

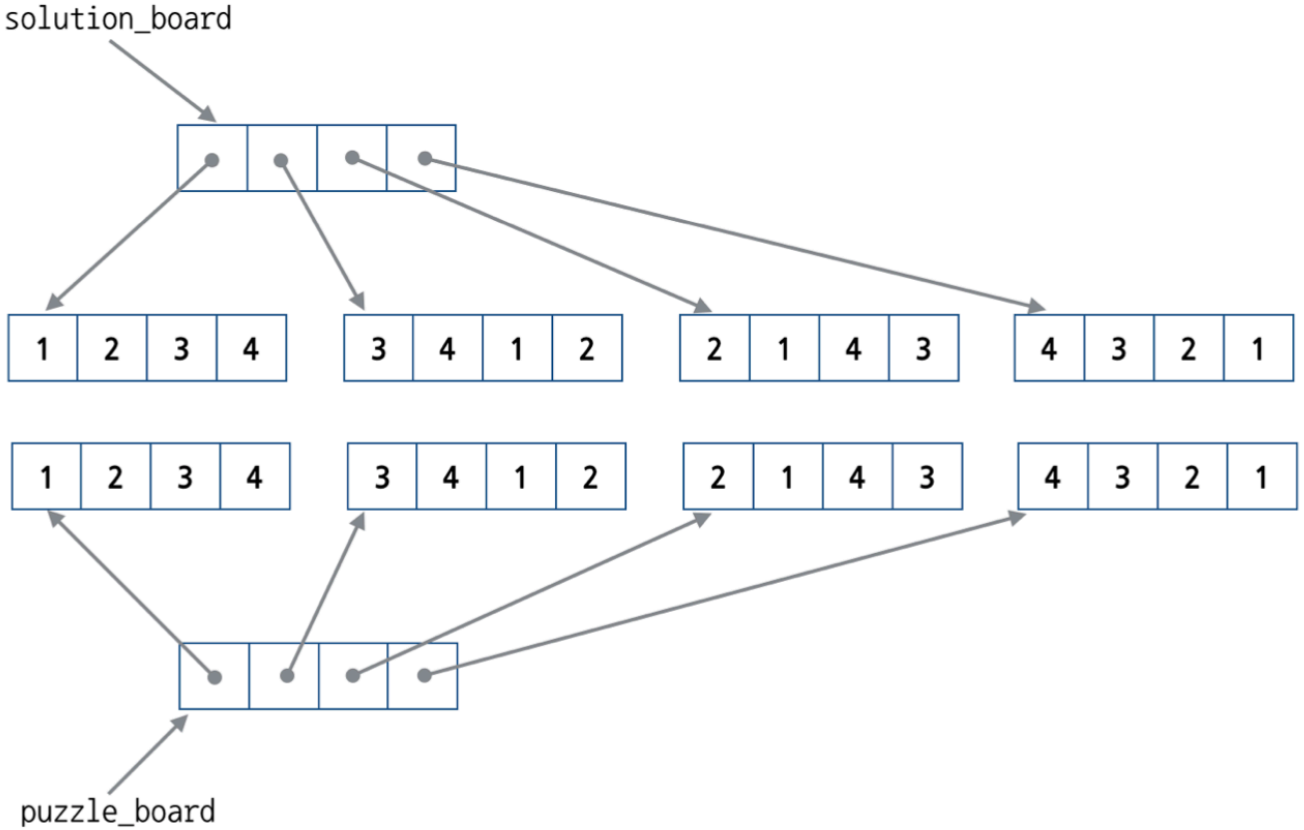
```
puzzle_board = solution_board[:]
```




```
1  def copy_board(board):
2      board_clone = []
3      for row in board:
4          board_clone.append(row[:])
5      return board_clone
```

퍼즐보드 복제하기

```
1  def copy_board(board):
2      board_clone = []
3      for row in board:
4          board_clone.append(row[:])
5      return board_clone
```



난이도 입력 받기

```
1   def get_level():
2       print("Enter your level.")
3       level = input("Beginner=1, Intermediate=2, Advanced=3: ")
4       while level not in ("1", "2", "3"):
5           level = input("Beginner=1, Intermediate=2, Advanced=3: ")
6           if level == "1":
7               return 6
8           elif level == "2":
9               return 8
10          else:
20              return 10
```

퍼즐보드 보여주기

```
1  def show_board(board):
2      for row in board:
3          for entry in row:
4              if entry == 0:
5                  print(".", end=" ")
6              else:
7                  print(entry, end=" ")
8          print()
```

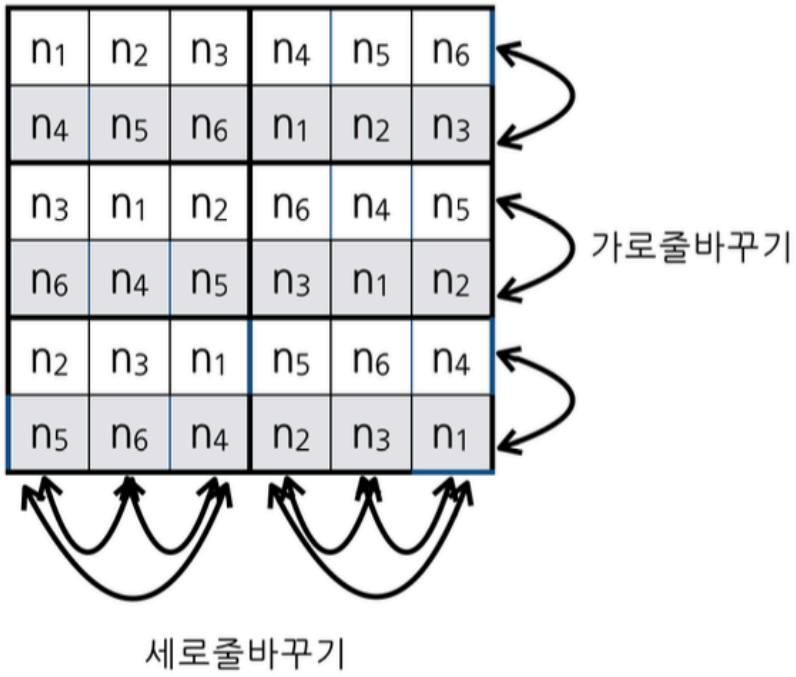
숫자 입력받기

```
1  def get_integer(message, i, j):
2      number = input(message)
3      while not (number.isdigit() and i <= int(number) <= j):
4          number = input(message)
5      return int(number)
```

난이도 입력 받기

```
1   def get_level():
2       print("Enter your level.")
3       level = input("Beginner=1, Intermediate=2, Advanced=3: ")
4       while level not in ("1", "2", "3"):
5           level = input("Beginner=1, Intermediate=2, Advanced=3: ")
6       if level == "1":
7           return 6
8       elif level == "2":
9           return 8
10      else:
20          return 10
```

프로젝트 옵션 #1: 6x6스도쿠



프로젝트 옵션 #2: 9x9스도쿠