

[Python]



Python으로 배우는

소프트웨어 원리

Appendix 07. 숫자 Puzzle 게임 개발

목차

1. 숫자 퍼즐 초기화
2. 숫자 퍼즐 위치 조작
3. 숫자 퍼즐 개선
4. 숫자 퍼즐 완성

01. 숫자 퍼즐 초기화

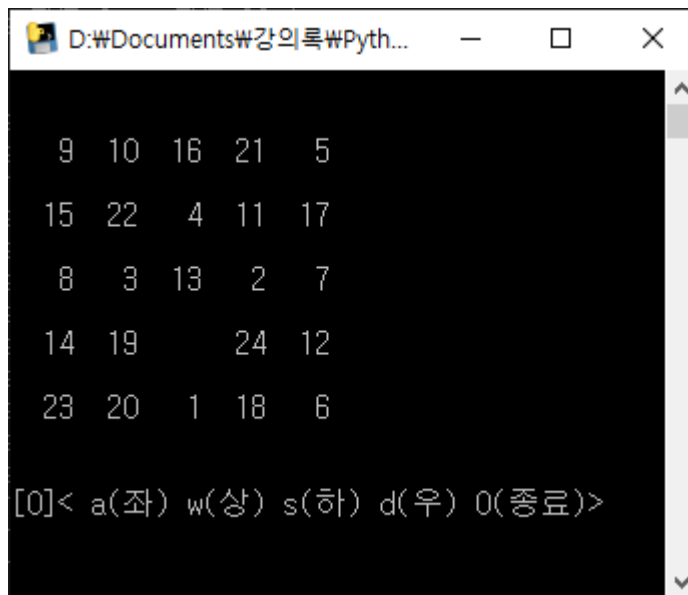
[Python실습]

● nxn 숫자 퍼즐 게임 만들기

Ch07-Puzzle.py

◆ 다음과 같은 숫자 퍼즐 정렬하기 게임을 완성 하시오.

- 퍼즐의 크기는 키보드로 입력 받음
- 퍼즐 이동 방향은 문자판 활용 → a:좌 w:상 s:하 d:우 0:종료 (빈 공간으로 주변 퍼즐이 이동하는 방향)
- 잘 못된 키 값이 입력되면 "??잘못된 키 값입니다!" 출력
- 이동할 수 없는 방향 값이 입력되면 "!!잘못된 이동입니다!" 출력
- 퍼즐 정렬이 완성되면 "%d번만에 성공하였습니다."를 출력하고 종료



```
D:\Documents\강의록\Pyth...  _  □  X

 9 10 16 21  5
15 22  4 11 17
 8  3 13  2  7
14 19    24 12
23 20  1 18  6

[0]< a(좌) w(상) s(하) d(우) 0(종료)>
```

I. 2차원 랜덤 배열 리스트 생성

Ch07-Puzzle01.py

◆ 2차원 배열 리스트를 중복없는 랜덤한 값으로 초기화 하시오.

- 초기 값은 0부터 시작하는 랜덤 수(중복 없이) 사용

```
import random
```

```
##전역변수 정의
```

```
row, col = (4, 4)
```

```
puzzle = []
```

```
arr = [i for i in range(row*col)]
```

```
arr = random.sample(arr, row*col)
```

```
for i in range(0, len(arr), col):
```

```
    puzzle.append(arr[i:i+col])
```

```
print(puzzle)
```

01. 숫자 퍼즐 초기화

[Python실습]

I. 2차원 랜덤 배열 리스트 생성

Ch07-Puzzle01.py

◆ 2차원 배열 리스트를 중복없는 랜덤한 값으로 초기화를 함수 호출로 하시오.

- 행(row)과 열(col)의 크기는 키보드로 입력받아서 사용
- 랜덤 퍼즐 배열 리스트를 생성하는 함수 호출로 해결 → **rand_puzzle()**
- 생성된 퍼즐을 입체적으로 보여주는 함수 호출로 해결 → **prt_puzzle()**
 - 0은 공백으로 출력

```
import random
```

```
##전역변수 정의
```

```
row, col = (4, 4)
```

```
puzzle = []
```

```
def rand_puzzle():
```

```
    global puzzle, row, col
```

```
##===== 메인 시작 =====##
```

```
## 변수 초기화
```

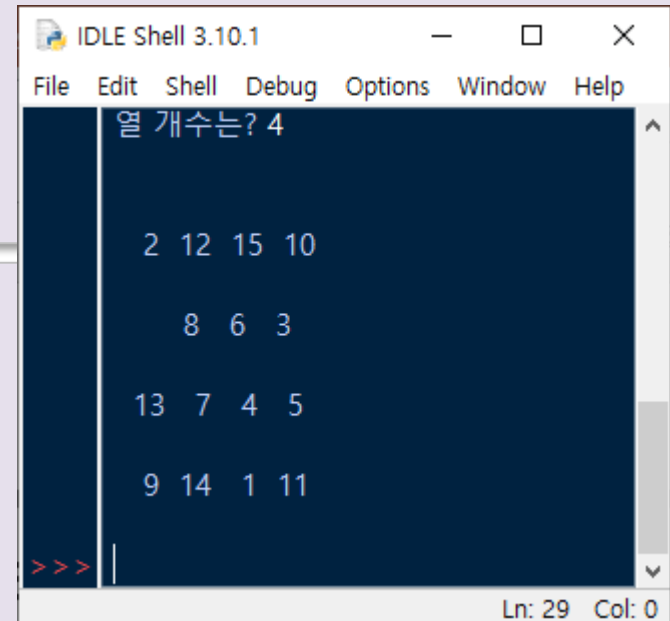
```
col = int(input("열 개수는? "))
```

```
row = col
```

```
rand_puzzle()
```

```
prt_puzzle()
```

```
##===== 메인 끝 =====##
```



```
IDLE Shell 3.10.1
File Edit Shell Debug Options Window Help
>>> 열 개수는? 4
2 12 15 10
8 6 3
13 7 4 5
9 14 1 11
>>>
```

02. 숫자 퍼즐 조작

[Python실습]

II. 퍼즐에서 0의 위치 값 찾기

Ch07-Puzzle02.py

◆ 2차원 배열 리스트에서 0의 위치 값을 구하시오.

• 0의 위치 값을 "row: %d, col: %d"로 출력

```
##===== 메인 시작 =====##
```

```
## 변수 초기화
```

```
col = int(input("열 개수는? "))
```

```
row = col
```

```
rand_puzzle()
```

```
prt_puzzle()
```

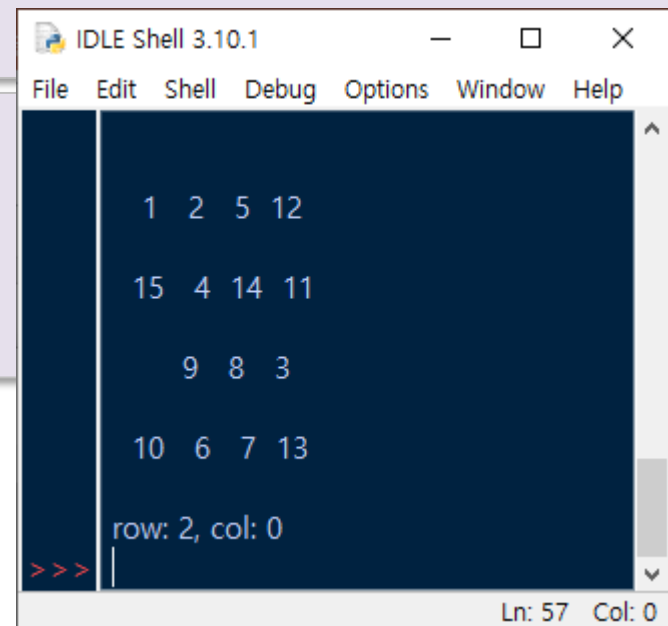
```
for i in range(row):
```

```
    for j in range(col):
```

```
        if puzzle[i][j] == 0:
```

```
            print("row: %d, col: %d" %(i, j))
```

```
##===== 메인 끝 =====##
```



The screenshot shows an IDLE Shell window titled "IDLE Shell 3.10.1". The window contains a 4x4 grid of numbers:

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

Below the grid, the text "row: 2, col: 0" is displayed. The shell prompt ">>>" is visible at the bottom left. The status bar at the bottom right shows "Ln: 57 Col: 0".

II. 퍼즐에서 0의 위치 값 찾기

Ch07-Puzzle02.py

◆ 2차원 배열 리스트에서 0의 위치 값을 함수 호출로 구하시오.

• 0의 위치 값을 함수 호출로 해결 → **find_zero()**

➢ 반환하는 0의 위치 값은 0의 순서 값으로 반환

➢ 예> 4x4에서 1행 2열은 6을 반환 → $seq = row * i + j$

• find_zero() 함수가 반환 한 순서 값(zero_seq)으로부터 0의 행(r)과 열(c) 값을 계산해서 출력

```
##===== 메인 시작 =====##
```

```
## 변수 초기화
```

```
col = int(input("열 개수는? "))
```

```
row = col
```

```
rand_puzzle()
```

```
prt_puzzle()
```

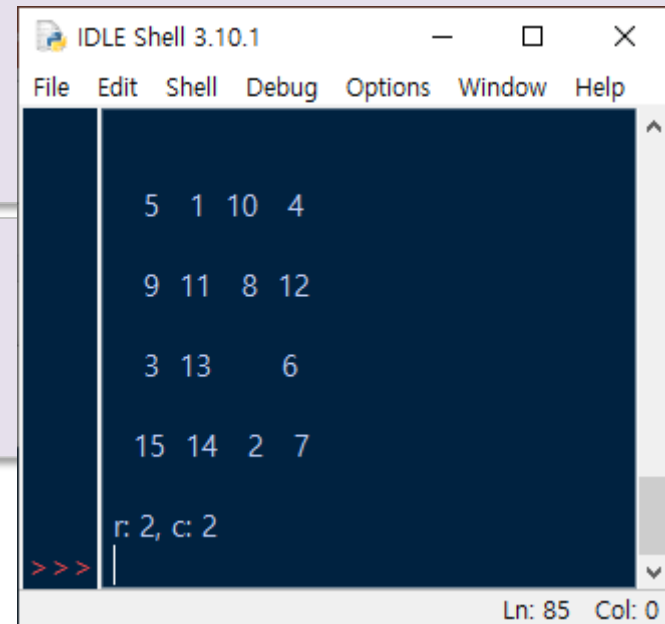
```
zero_seq = find_zero()
```

```
r = zero_seq // row # zero 위치 행
```

```
c = zero_seq % row # zero 위치 열
```

```
print("r: %d, c: %d" %(r, c))
```

```
##===== 메인 끝 =====##
```



The screenshot shows the IDLE Shell 3.10.1 window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main area displays a 4x4 grid of numbers:

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

Below the grid, the output of the find_zero() function is shown: r: 2, c: 2. The prompt >>> is visible at the bottom left. The status bar at the bottom right indicates Ln: 85 Col: 0.

02. 숫자 퍼즐 조작

[Python실습]

III. 키보드로 방향키 값 받아 퍼즐 이동

Ch07-Puzzle03.py

◆ 키보드로 입력 받은 퍼즐 방향 문자를 출력하시오.

- 퍼즐 이동 방향은 문자판 활용 → a:좌 w:상 s:하 d:우 0:종료 (빈 공간으로 주변 퍼즐이 이동하는 방향)
- 잘 못된 키 값이 입력되면 "??잘못된 키 값입니다!" 출력

```
##===== 메인 시작 =====##  
## 변수 초기화  
col = int(input("열 개수는? "))  
row = col  
  
rand_puzzle()  
prt_puzzle()
```

```
while True :  
    key = input("a(좌) w(상) s(하) d(우) > ")  
    if key == "a":  
        print("Left")  
        :  
        :  
    elif key == "0":  
        print("End")  
    else:  
        print("??잘못된 키 값입니다!")  
##===== 메인 끝 =====##
```


III. 키보드로 방향키 값 받아 퍼즐 이동

Ch07-Puzzle03.py

◆ 입력 받은 방향으로 퍼즐 이동 결과를 리스트에 반영하여 출력하시오.

•퍼즐 이동 방향은 문자판 활용 → a:좌 w:상 s:하 d:우 0:종료 (빈 공간으로 주변 퍼즐이 이동하는 방향)

```
while True :
    zero_seq = find_zero()
    r = zero_seq // row # zero 위치 행
    c = zero_seq % row # zero 위치 열
    key = input("a(좌) w(상) s(하) d(우) > ")
    if key == "a": #LEFT
        puzzle[r][c], puzzle[r][c] = puzzle[r][c], puzzle[r][c] #0 우측 값과 좌측 0과 값 교환
    elif key == "d": #RIGHT
        puzzle[r][c], puzzle[r][c] = puzzle[r][c], puzzle[r][c] #0 좌측 값과 우측 0과 값 교환
    elif key == "s" : #DOWN
        puzzle[r][c], puzzle[r][c] = puzzle[r][c], puzzle[r][c] #0 위쪽 값과 아래 0과 값 교환
    elif key == "d": #UP
        puzzle[r][c], puzzle[r][c] = puzzle[r][c], puzzle[r][c] #0 아래쪽 값과 위 0과 값 교환
    elif key == "0":
        break
    else:
        print("??잘못된 키 값입니다!")
    prt_puzzle()
##===== 메인 끝 =====##
```

III. 키보드로 방향키 값 받아 퍼즐 이동

Ch07-Puzzle03.py

◆ 입력 받은 방향으로 퍼즐 이동 결과를 리스트에 반영하여 출력하시오.

• 퍼즐의 0 위치 값 (r, c)을 함수로 전달하여 리스트에 인접 값 간에 교환 처리

➤ 만약 상호 교환할 수 없는 위치 값이면 1을 반환, 교환 가능하면 교환 후 0을 반환

```
def left(r, c):  
    global puzzle, col  
    if c+1 >= col:  
        print("이동 불가!")  
    else:  
        puzzle[r][c+1], puzzle[r][c] =
```

```
while True :  
    zero_seq = find_zero()  
    r = zero_seq // row # zero 위치 행  
    c = zero_seq % row # zero 위치 열  
    key = input("a(좌) w(상) s(하) d(우) > ")  
    if key == "a": #LEFT  
        left(r, c)  
    if key == "d": #RIGHT  
        right(r, c)  
    if key == "s" : #DOWN  
        down(r, c)  
    if key == "w": #UP  
        up(r, c)  
    elif key == "0": #END  
        break  
    else:  
        print("??잘못된 키 값입니다!")  
    prt_puzzle()  
##===== 메인 끝 =====##
```

IV. 스크린 리프레시 후 퍼즐 출력

Ch07-Puzzle04.py

◆ 이전 퍼즐 내용을 지우고 변경된 퍼즐을 다시 출력하기

- 이전 퍼즐 내용을 스크린 클리어 기능으로 클리어 시킴

```
import os
```

```
while True :
    zero_seq = find_zero()
    r = zero_seq // row # zero 위치 행
    c = zero_seq % row # zero 위치 열
    key = input("a(좌) w(상) s(하) d(우) > ")
    if key == "a":      #LEFT
        left(r, c)
        :
        :
    elif key == "0":    #END
        break
    else:
        print("??잘못된 키 값입니다!")

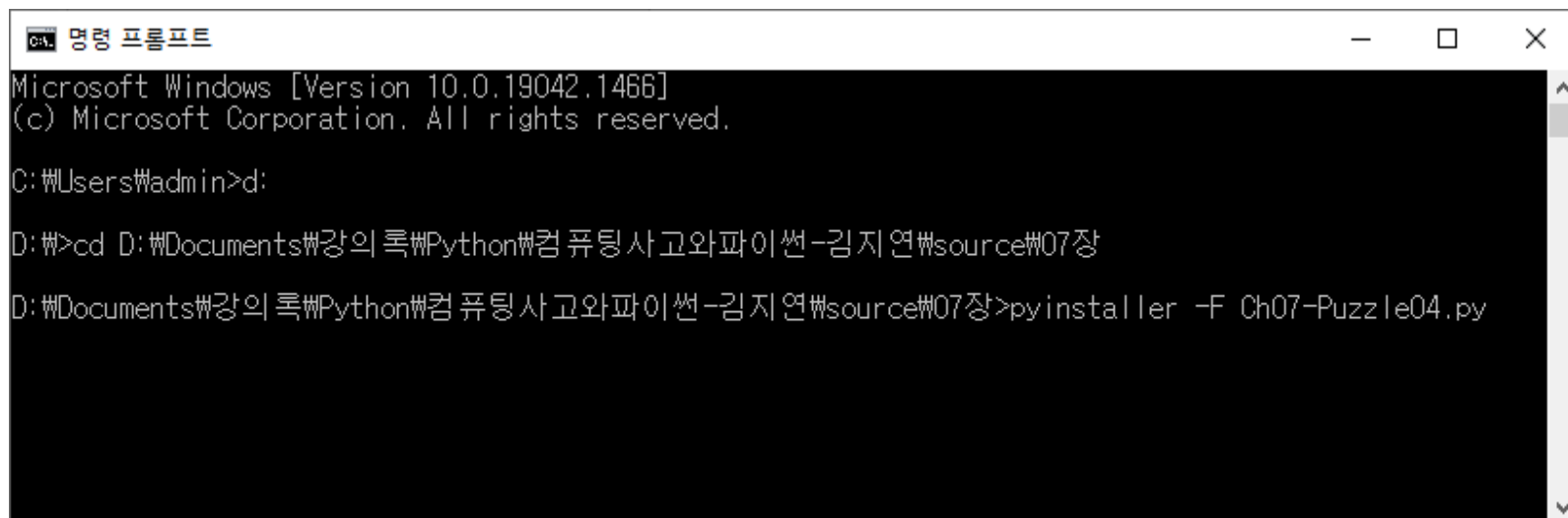
os.system('cls')      #os.system('cls' if os.name == 'nt' else 'clear')
prt_puzzle()
```

IV. 스크린 리프레시 후 퍼즐 출력

Ch07-Puzzle04.py

◆ 컴파일 하여 실행파일을 만들어서 결과 확인하기

- 명령 프롬프트 실행
- Ch07-Puzzle04.py가 있는 위치로 이동 → (예) `cd D:\project\Ch07`
- 컴파일 수행 실행파일 생성 → `pyinstaller -F Ch07-Puzzle04.py`
- 실행 파일 실행
 - `cd dist` → 실행 파일 생성 위치로 이동
 - `Ch07-Puzzle04.exe` → 실행 파일 실행



```
CA 명령 프롬프트
Microsoft Windows [Version 10.0.19042.1466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin>d:

D:\>cd D:\Documents\강의록\Python\컴퓨팅사고와파이썬-김지연\source\07장
D:\Documents\강의록\Python\컴퓨팅사고와파이썬-김지연\source\07장>pyinstaller -F Ch07-Puzzle04.py
```

V. 엔터 없이 키 값 처리

Ch07-Puzzle05.py

◆ 방향 문자 입력 시 문자 입력 후 바로 처리되도록 개선

```
import msvcrt

LEFT, DOWN, RIGHT, UP, END = ('LEFT', 'DOWN', 'RIGHT', 'UP', 'END')
arrow_keys = {                #[딕셔너리 자료형] 키보드 입력 값(확장자 b 포함)을 키 값으로 대응
    b'w': UP,
    b's': DOWN,
    b'd': RIGHT,
    b'a': LEFT,
    b'0': END
}

while True :
    :
    #key = input("a(좌) w(상) s(하) d(우) > ")
    print("< a(좌) w(상) s(하) d(우) 0(종료)> ")
    ch = msvcrt.getch()
    if ch in arrow_keys.keys():
        key = arrow_keys[ch]
    else:
        key = 'Wrong'

    if key == 'LEFT' :
        left(r, c)
```

VI. 퍼즐 완성 여부 판단

Ch07-Puzzle06.py

◆ 2차원 배열 리스트 정렬이 완성되었는지 판단을 함수 호출로 해결하시오.

- 전체 중에 0을 제외한 숫자가 모두 일치하는지 판단을 함수 호출로 해결 → **check_complete()**
- 정렬이 완성되었으면 1을 반환, 미완성이면 0을 반환

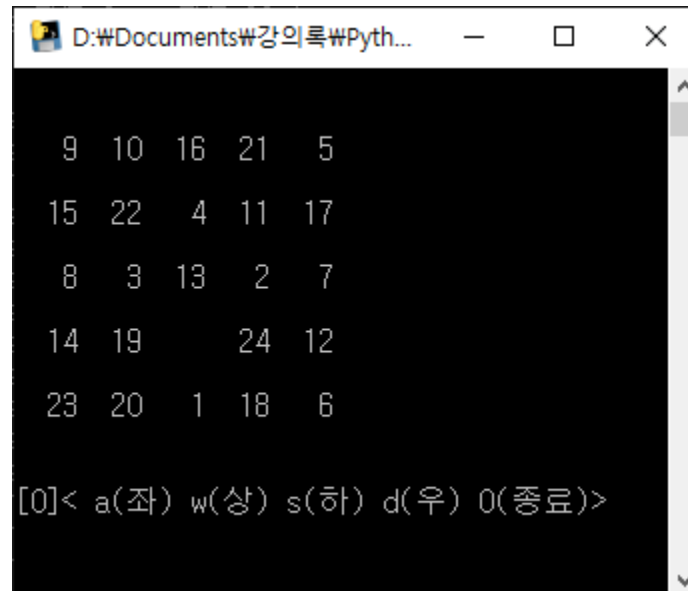
```
while True :  
    :  
    :  
  
    os.system('cls')    #os.system('cls' if os.name == 'nt' else 'clear')  
    prt_puzzle()  
  
    if check_complete() :  
        print(">성공했습니다.")  
    else :  
        print(">미완성입니다.")  
##===== 메인 끝 =====##
```

04. 숫자 퍼즐 완성

● [과제] nxn 숫자 퍼즐 게임 만들기

◆ 다음 사항을 추가하여 숫자 퍼즐 게임을 완성하십시오.

- 잘 못된 키 값('a', 'w', 's', 'd' 이외)이 입력되면 "??잘못된 키 값입니다!"를 출력
- 이동할 수 없는 방향 값이 입력되면 "!!잘못된 이동입니다!"를 출력
- 퍼즐 정렬이 완성되면 "%d번만에 성공하였습니다."를 출력하고 종료
- 현재 시도 횟 수를 아래와 같이 좌측 하단에 표시
→ 잘 못된 키 값이거나 이동할 수 없는 방향 값일 경우는 회 수에서 제외



```
D:\Documents\강의록\Pyth...  
  
9 10 16 21 5  
15 22 4 11 17  
8 3 13 2 7  
14 19 24 12  
23 20 1 18 6  
  
[0]< a(좌) w(상) s(하) d(우) 0(종료)>
```

Thank You !

[Python]