

CSED101. Programming & Problem solving

Spring 2023

Programming Assignment #2

김정우 (jwk00216@postech.ac.kr)

■ 제출 마감일: 2023.05.08 23:59

■ 파이썬 버전: Python 3.x

■ 제출물

- .py 소스 코드 (assn2.py)
 - 프로그램의 소스 코드에 채점자의 이해를 돕기 위한 주석을 반드시 붙여주세요.
- 보고서 파일 (.docx, .hwp 또는 .pdf; assn2.docx, assn2.hwp 또는 assn2.pdf)
 - 보고서는 AssnReadMe.pdf를 참조하여 작성하시면 됩니다.
 - 명예 서약 (Honor code): 표지에 다음의 서약을 기입하여 제출해 주세요: “나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.” 보고서 표지에 명예 서약이 기입되어 있지 않은 과제는 제출되지 않은 것으로 처리됩니다.
 - 작성한 소스 코드와 보고서 파일은 PLMS를 통해 제출해 주세요.

■ 주의 사항

- 구문 오류(Syntax Error)가 발생하거나 실행이 되지 않는 과제는 0점으로 채점됩니다.
- 제출 기한보다 하루 늦게 제출된 과제는 최종 20%, 이틀 늦게 제출된 과제는 최종 40% 감점됩니다. 제출 기한보다 사흘 이상 늦으면 제출 받지 않습니다 (0점 처리).
늦은 제출시 PLMS에 기록된 최종 수정일시를 기준으로 감점합니다.
- 각 문제의 제한 조건과 요구 사항을 반드시 지켜 주시기 바랍니다.
- 모든 문제의 출력 형식은 채점을 위해 아래에 제시된 예시들과 최대한 비슷하게 작성해 주세요.
- 각 문제에 명시된 예외 처리 외에는 고려하지 않아도 됩니다.
- 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.
- 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 “POSTECH 전자컴퓨터공학부 부정행위 정의”를 따릅니다 (PLMS의 본 과목 공지사항에 등록된 글 중, 제목이 [document about cheating]인 글에 첨부되어 있는 disciplinary.pdf를 참조하세요).

■ Problem: 슬라이딩 퍼즐 (Sliding Puzzle)

[문제]

슬라이딩 퍼즐을 맞추는 프로그램을 작성해봅니다. 슬라이딩 퍼즐은 그림 1 과 같이 24(5x5 size 일 때) 개의 퍼즐 블록과 하나의 빈칸으로 구성되어 있습니다. 퍼즐을 맞추기 전 상태가 그림 1의 (a)라고 했을 때, 빈칸인 *를 이동시켜가면서 그림 1의 (b)의 상태로 만들면 성공하는 퍼즐입니다.

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

(a) 슬라이딩 퍼즐의 초기 상태

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

(b) 슬라이딩 퍼즐의 최종 목표

<그림 1. 슬라이딩 퍼즐의 예제>

[목적]

- 2차원 리스트 사용을 익힙니다.
- 조건문, 반복문 사용을 익힙니다.
- 사용자 정의 함수 및 랜덤 모듈 함수 사용법을 익힙니다.

[주의사항]

- (1) 소스코드 저장 시 이름은 `assn2.py` 로 작성합니다.
- (2) 보고서는 `assn2.docx`, `assn2.hwp` 또는 `assn2.pdf` 로 저장 합니다.
- (3) 이번 과제는 2차원 리스트 사용을 위한 과제입니다. 퍼즐은 반드시 2차원 리스트로 생성 후, 사용하도록 합니다.
- (4) 문서에 반드시 정의해서 사용해야 할 사용자 정의 함수가 설명되어 있으니 확인 후 구현하도록 합니다.
- (5) `int`, `float`, `bool`, `str`, `list`, `tuple`, `dictionary` 자료형까지 수업시간에 다룬 내용에 한해서 사용 가능합니다.
- (6) 퍼즐 화면을 그리기 위해 문자는 `-`, `+`, `|` 를 사용하세요.

[설명 및 요구사항]

1. 퍼즐 생성

프로그램을 실행하면 아래의 조건을 만족하는 퍼즐을 생성하여 출력합니다.

- 퍼즐의 크기는 $n \times n$ 이며, 2차원 리스트로 구현합니다.
- 사용자로부터 3이상 6이하의 정수 n 을 입력 받아 퍼즐을 생성합니다.
- 퍼즐에는 $1 \sim n^2-1$ 까지의 정수와 빈칸을 의미하는 * 문자가 들어갑니다
 - Random 모듈을 사용하여 매 실행시마다 랜덤하게 초기화된 퍼즐이 출력되도록 합니다.
- 퍼즐에는 중복된 값이 없어야 합니다.

2. 게임 시작 화면

프로그램을 실행하면 아래 예시처럼, Choose the puzzle size 문구가 출력됩니다. 3이상 6이하의 정수를 입력하여 퍼즐 크기를 선택하면, Enjoy the puzzle 문구와 함께 퍼즐이 출력됩니다. (예시의 빨간색 밑줄은 사용자 입력을 말함)

```
Choose the puzzle size: 7
Wrong size!
Choose the puzzle size: 5
Enjoy the puzzle!
+---+---+---+---+---+
| 4 | 6 | 7 | 21 | 5 |
+---+---+---+---+---+
| 1 | 22 | 17 | 2 | 3 |
+---+---+---+---+---+
| 9 | 12 | 14 | 19 | 20 |
+---+---+---+---+---+
| 24 | * | 10 | 15 | 11 |
+---+---+---+---+---+
| 8 | 13 | 23 | 16 | 18 |
+---+---+---+---+---+
Enter the command ... (h: help, q: quit):
```

<그림 2. 슬라이딩 퍼즐 프로그램 실행 첫 화면 예시>

퍼즐 크기 입력 시, 범위(3이상 6이하)를 벗어난 입력의 경우 "Wrong size!" 문구와 함께 다시 입력 받습니다.

3. 명령어

본 퍼즐에서 사용할 수 있는 명령어는 총 9 가지이며, 각 명령어의 이름과 해당되는 기능은 아래 그림 3에서 확인할 수 있습니다.

3.1. 명령어 h

h를 입력하면 아래와 같이 명령어의 종류와 기능에 대한 설명을 출력합니다.

```
Enter the command ... (h: help, q: quit): h
w: Move * to UPPER block
s: Move * to LOWER block
a: Move * to LEFT block
d: Move * to RIGHT block
c: use CHANCE
n: Create NEW puzzle (randomly generated puzzle)
f: Create NEW puzzle (import puzzle from txt file)
h: Print help message
q: Quit the puzzle

Enter the command ... (h: help, q: quit): wrong
Wrong input!

Enter the command ... (h: help, q: quit):
```

<그림 3. h명령어 사용 예시>

위 명령어 외 입력 시, 위의 예시처럼 "Wrong input!"이라는 문구와 함께 명령어를 다시 입력 받습니다.

3.2. w(상), s(하), a(좌), d(우) 이동 키



w, s, a, d는 각각 상,하,좌,우 로 빈 블록(*)을 이동시키는 명령어입니다.

키를 입력하여 이동 명령어를 내리면 해당 방향으로 빈 블록(*)을 이동시킨 후 변환된 퍼즐을 다시 출력합니다. 그 후 Enter the command ... 명령어를 출력하고 다음 명령어를 기다립니다.

해당 방향으로 더 이상 빈 블록(*)을 옮길 수 없다면, “Can not move the *” 문구를 출력하고, 다시 Enter the command ... 문구를 출력하며 다음 명령어를 기다립니다.

아래는 d 키를 입력하여 오른쪽으로 한 칸 움직인 예시입니다.

```

Enjoy the puzzle!
+---+---+---+---+
| 17 | 8 | 12 | 6 | 16 |
+---+---+---+---+
| 22 | 7 | 14 | * | 10 |
+---+---+---+---+
| 19 | 20 | 24 | 23 | 15 |
+---+---+---+---+
| 2 | 5 | 18 | 13 | 9 |
+---+---+---+---+
| 11 | 4 | 3 | 21 | 1 |
+---+---+---+---+
Enter the command ... (h: help, q: quit): d

+---+---+---+---+
| 17 | 8 | 12 | 6 | 16 |
+---+---+---+---+
| 22 | 7 | 14 | 10 | * |
+---+---+---+---+
| 19 | 20 | 24 | 23 | 15 |
+---+---+---+---+
| 2 | 5 | 18 | 13 | 9 |
+---+---+---+---+
| 11 | 4 | 3 | 21 | 1 |
+---+---+---+---+
Enter the command ... (h: help, q: quit): d
Can not move the *

+---+---+---+---+
| 17 | 8 | 12 | 6 | 16 |
+---+---+---+---+
| 22 | 7 | 14 | 10 | * |
+---+---+---+---+
| 19 | 20 | 24 | 23 | 15 |
+---+---+---+---+
| 2 | 5 | 18 | 13 | 9 |
+---+---+---+---+
| 11 | 4 | 3 | 21 | 1 |
+---+---+---+---+
Enter the command ... (h: help, q: quit):

```

<그림 4. 이동 명령어 사용 예시>

3.3. Chance 사용(c)

명령어 `c`를 입력하면 찬스를 사용할 수 있습니다. 찬스를 사용하면, 두 블록의 위치에 관계없이 좌표를 입력 받아 위치를 맞바꿀 수 있습니다. 아래 그림은 명령어 `c`를 입력하였을 때의 예시입니다.

```
+-----+
| 4 | 6 | 7 | 21 | 5 |
+-----+
| 1 | 22 | 17 | 2 | 3 |
+-----+
| 9 | 12 | 14 | 19 | 20 |
+-----+
| 24 | * | 10 | 15 | 11 |
+-----+
| 8 | 13 | 23 | 16 | 18 |
+-----+
Enter the command ... (h: help, q: quit): c
Use CHANCE! Choose the blocks: 01 23

+-----+
| 4 | 19 | 7 | 21 | 5 |
+-----+
| 1 | 22 | 17 | 2 | 3 |
+-----+
| 9 | 12 | 14 | 6 | 20 |
+-----+
| 24 | * | 10 | 15 | 11 |
+-----+
| 8 | 13 | 23 | 16 | 18 |
+-----+
Enter the command ... (h: help, q: quit):
```

<그림 5. c명령어 사용 예시>

위에서는 `c`를 입력한 후 좌표를 입력 받아, (0,1)에 위치한 6과 (2,3)에 위치한 19를 교환했습니다. 이때 빈 블록(*) 또한 이동시킬 수 있습니다.

입력 받은 좌표 값이 퍼즐의 범위를 벗어나거나, 정수가 아닌 입력이 들어올 경우 `Wrong input!` 메시지와 함께 좌표를 다시 입력 받습니다.

3.4. 퍼즐 생성 명령어

명령어 `n` 과 `f` 는 모두 새로운 퍼즐을 생성하는 명령어입니다. 명령어 `n` 은 퍼즐을 랜덤하게 생성하고 (randomly defined puzzle), `f` 는 외부 파일을 읽어서 퍼즐을 생성하게 됩니다 (import puzzle from txt file).

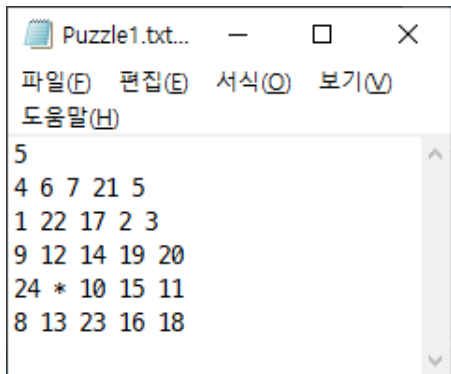
아래 그림은 명령어 `n` 을 입력하였을 때의 예시입니다.

```
+---+---+---+---+---+
| 11 | 4 | 3 | 21 | 1 |
+---+---+---+---+
Enter the command ... (h: help, q: quit): n
Choose the puzzle size: 5
Enjoy the puzzle!
+---+---+---+---+---+
| 24 | 11 | 17 | 14 | 3 |
+---+---+---+---+---+
| 2 | 8 | 15 | 9 | 12 |
+---+---+---+---+---+
| 6 | 16 | 23 | 4 | 10 |
+---+---+---+---+---+
| 1 | * | 21 | 18 | 13 |
+---+---+---+---+---+
| 20 | 7 | 22 | 19 | 5 |
+---+---+---+---+---+
Enter the command ... (h: help, q: quit):
```

<그림 6. n 명령어 사용 예시>

명령어 `f` 를 입력하면 외부 파일을 읽어 퍼즐을 생성할 수 있습니다. 이때 사용자로부터 파일명을 입력 받아, 해당 파일을 읽어서 퍼즐을 생성합니다.

아래 예시는 Puzzle1.txt 를 읽어 퍼즐을 생성한 예시입니다.



```
Enter the command ... (h: help, q: quit): f
File Name: Puzzle1.txt
Enjoy the puzzle!
+---+---+---+---+---+
| 4 | 6 | 7 | 21 | 5 |
+---+---+---+---+---+
| 1 | 22 | 17 | 2 | 3 |
+---+---+---+---+---+
| 9 | 12 | 14 | 19 | 20 |
+---+---+---+---+---+
| 24 | * | 10 | 15 | 11 |
+---+---+---+---+---+
| 8 | 13 | 23 | 16 | 18 |
+---+---+---+---+---+
Enter the command ... (h: help, q: quit):
```

<그림 7. f 명령어 사용 예시 - 유효한 파일>

이때 파일은 텍스트(.txt) 파일이며, 파일의 구성은 다음과 같습니다.

- 파일의 첫 번째 줄의 숫자(`n`)는 퍼즐 크기(`n x n`)를 나타냅니다.
- 두 번째 줄부터 퍼즐의 각 행에 해당하는 정보가 줄(line) 단위로 구분되어 있습니다.

그림 7의 Puzzle1.txt 는 퍼즐 크기가 5 x 5 에 해당하며 퍼즐의 1 행부터 5 행에 해당하는 정보가 파일의 두 번째 줄부터 순서대로 저장되어 있음을 볼 수 있습니다.

사용자로부터 파일명을 입력 받으면, 게임에 유효한 퍼즐인지 체크해서 아닌 경우에는 "Error!" 문구 출력 후, 다시 파일명을 입력 받습니다.

다음 중 하나라도 해당되면 유효하지 않은 것으로 간주합니다. 단, 파일의 첫 번째 줄은 항상 유효한 정수(3~6)가 들어가 있다고 가정합니다.

- 1) 블록의 값이 $1 \sim n^2 - 1$ 이 아닌 다른 값이 있는 경우
- 2) 중복되는 블록의 값이 존재하는 경우
- 3) 빈 블록(*)이 없는 경우
- 4) 행의 개수가 n 개가 아닌 경우
- 5) 하나의 행(row)에 블록 개수가 n 개가 아닌 경우

Puzzle-w1.txt	Puzzle-w2.txt	Puzzle-w3.txt	Puzzle-w4.txt	Puzzle-w5.txt
3 1 2 * 3 4 5 6 7 11	3 * 2 3 4 5 6 7 8 8	4 1 2 3 4 6 7 8 9 10 11 12 13 14 15 *	4 15 14 13 12 11 10 9 8 * 7 6 5	3 1 2 3 4 5 6 7 8 9 * 10 11 12 13 14 15
1)에 해당	2)에 해당	5)에 해당	4)에 해당	1), 4), 5) 에 해당

(a) 유효하지 않은 텍스트 파일 예시

```

Enter the command ... (h: help, q: quit): f
File Name: Puzzle-w1.txt
Error!
File Name: Puzzle-w2.txt
Error!
File Name:
  
```

(b) f 명령어 사용 예시 - 유효하지 않은 파일

<그림 8. 유효하지 않은 텍스트 파일과 f 명령어 사용 예시>

파일로부터 완성된 퍼즐을 읽은 경우, 그림 10의 예시처럼 "The puzzle is solved." 문구 출력 후, 프로그램을 종료합니다.

- 1) 파일에 숫자와 * 이외의 문자가 포함되는 경우는 고려하지 않습니다.
- 2) 경로내에 파일이 존재하지 않는 경우, "No file is found" 문구 출력과 함께 파일명을 다시 입력 받습니다.

3.5. 명령어 q

q를 입력하면 프로그램을 종료합니다. 아래 예시처럼 "Quit the puzzle game." 문구 출력과 함께 프로그램을 종료합니다.

```

Enter the command ... (h: help, q: quit): q

Quit the puzzle game.
  
```

<그림 9. q명령어 사용 예시>

4. 프로그램 종료

명령어 q를 입력하거나, 퍼즐 맞추기에 성공하면 프로그램을 종료합니다.

퍼즐 맞추기에 성공한 경우, 완성된 퍼즐을 출력한 후 "The puzzle is solved." 문구 출력과 함께 프로그램을 종료합니다.

```
+---+---+---+---+---+
| 1 | 2 | 3 | 4 | 5 |
+---+---+---+---+---+
| 6 | 7 | 8 | 9 | 10 |
+---+---+---+---+---+
| 11 | 12 | 13 | 14 | 15 |
+---+---+---+---+---+
| 16 | 17 | 18 | 19 | 20 |
+---+---+---+---+---+
| 21 | 22 | 23 | * | 24 |
+---+---+---+---+---+
Enter the command ... (h: help, q: quit): q

+---+---+---+---+---+
| 1 | 2 | 3 | 4 | 5 |
+---+---+---+---+---+
| 6 | 7 | 8 | 9 | 10 |
+---+---+---+---+---+
| 11 | 12 | 13 | 14 | 15 |
+---+---+---+---+---+
| 16 | 17 | 18 | 19 | 20 |
+---+---+---+---+---+
| 21 | 22 | 23 | 24 | * |
+---+---+---+---+---+
The puzzle is solved.
```

<그림 10. 프로그램 종료 예시(퍼즐 완성)>

[사용자 정의 함수]

다음 함수들을 반드시 작성하여 프로그램을 구현해야 합니다. 아래 명시된 함수 이름은 변경하지 말아주세요. 각 함수의 매개변수의 개수 및 리턴 값 등은 자유롭게 변경 가능 합니다. 그러나 동일한 기능을 하는 함수는 반드시 있어야 하며, 변경 시 무엇을 어떻게 변경했는지 보고서에 기록 하도록 합니다. 이 외에 필요한 함수는 정의해서 사용할 수 있습니다.

- `print_puzzle(puzzle, size)`
 - 생성한 퍼즐(puzzle)과 퍼즐 크기(size)를 매개변수로 전달 받아, 아래 예시와 같이 퍼즐의 현재 상태를 출력하는 함수입니다.
 - 게임 시작화면과 더불어 게임 진행 과정 중에 계속 호출하여 사용하도록 합니다.

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

- `find_blank(puzzle, size)`
 - 퍼즐에서 빈 블록(*)의 위치를 찾아서 좌표를 반환하는 함수입니다.
위 예시의 경우 (4, 3)을 반환 합니다.
- `swap_blocks(puzzle, block_a, block_b)`
 - 블록 a의 좌표(block_a)와 블록 b의 좌표(block_b)를 매개변수로 전달받아, 퍼즐에서 블록 a와 블록 b의 값을 서로 교환합니다.
 - 블록 이동 시, 해당 함수를 호출하여 사용합니다.
- `move_blank_block(puzzle, size, command, blank)`
 - 이동 명령어(command)와 빈 블록(*)의 좌표(blank)를 매개변수로 전달받아, 이동 명령어(w, s, a, d)에 맞게 움직이는 함수입니다.
- `create_random_puzzle(size)`
 - 새로운 퍼즐(2차원 리스트)을 랜덤하게 생성하여 반환하는 함수입니다.
 - 프로그램 시작과 명령어 n 을 받았을 때, 호출하여 사용하도록 합니다.
- `create_from_file(filename)`
 - 파일명을 매개변수로 전달 받아 해당 파일의 퍼즐을 생성하여 반환하는 함수입니다.
 - 명령어로 f를 받았을 때 호출하여 사용하도록 합니다.
- `is_valid(puzzle, size):`
 - 퍼즐이 유효한지 알려주는 함수입니다. 7페이지에 명시되어 있는 유효한 퍼즐 조건을 만족할 경우에는 True, 그렇지 않을 경우에는 False를 반환합니다. 명령어로 f를 입력 받은 후 파일명을 입력 받았을 때 퍼즐이 유효한지 확인하기 위해 사용하도록 합니다.
- `is_solved(puzzle, size)`
 - 퍼즐이 풀렸는지 알려주는 함수입니다. 사용자가 퍼즐을 풀었을 경우 True, 그렇지 않을 경우 False를 반환합니다. 매 시점 is_solved 함수를 호출하여 퍼즐 종료 여부를 확인합니다.