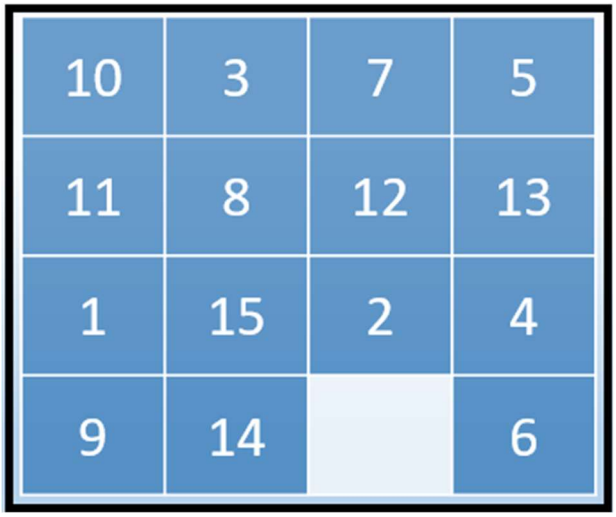
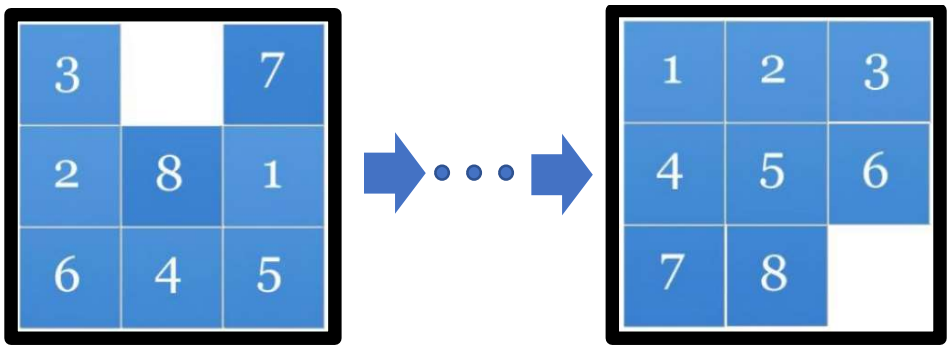
Sliding Puzzle - 2020



OVERVIEW

In this assignment, you are going to design and develop an interactive sliding puzzle game for both 8 and 15 numbers. For an 8-number puzzle, it has a square-framed board consisting of 8 square tiles, numbered 1 to 8, initially placed in random order, while a 15-number puzzle there are 15 numbered square tiles, from 1 to 15, as shown above. The board has an empty space where an adjacent tile can be slid to. The objective of the game is to re-arrange the tiles into a sequential order by their numbers (left to right, top to bottom) by repeatedly making sliding moves (left, right, up or down). The following figure shows an example of an 8-number puzzle where "INITIAL" is the starting point of the game, and the player needs to repeatedly slide an adjacent tile, one at a time, to the currently unoccupied space (the empty space) until all numbers appear sequentially, ordered from left to right, top to bottom, shown as "FINAL".



SCOPE

1. At the start of the game, display a brief introduction about the game.
2. Prompt user to enter the 4 letters used for the left, right, up and down moves. User is free to pick any four letters such as j, k, r, f for left, right, up and down respectively.
3. Prompt user for selection of either 8 or 15 puzzle, then generate a randomized, SOLVABLE 8 or 15-tile puzzle accordingly; have it displayed on the screen using simple ASCII characters.
4. Repeatedly prompt player the sliding direction (left, right, up or down) - the direction that the adjacent tile to be moved (not the empty location). In the prompt the valid sliding move(s) are shown together with the designated letter (from step 2 above). For example:
    a. Enter your move (left-j, right-k) >
5. After each move, show the updated puzzle on the screen and prompt further direction if needed.
6. Inform user when the puzzle is solved (i.e. the numbered tiles are in sequential order, left to right, top to bottom); then prompt user to continue another game or end the program.
7. Track total number of moves made for each game and have it displayed as the puzzle is solved.

NOTE:

- Keep your entire source code in ONE SINGLE file.
- Use only standard python modules
- In your design stick ONLY to functions, in other words, no class objects of your own.

STARTUP OPTIONS

Not applicable

SKILLS

In this assignment, you will be trained on the use of the followings:

- Use input() to prompt user for information
- Use standard objects (strings, numbers & lists)
- Control statements to interact with users
- Variable Scope
- String formatting (method style)
- Functions (with parameters and return) for program structure and decomposition

DELIVERABLES

1. Design documentation (A1_School_StudentID_Design.doc/pdf)
2. Program source code (A1_School_StudentID_Source.py)

where School is SSE, SME, HSS, FE or LHS and StudentID is your 9-digit student ID.

Zip all files above in a single file (A1_School_StudentID.zip) and submit the zip file by due date to the corresponding assignment folder under "Assignment (submission)"

For instances, a SME student with student ID "119010001":

- A1_SME_119010001.zip:
  - A1_SME_119010001_Design.doc/pdf
  - A1_SME_119010001_Source.py

5% will be deducted if any files are incorrectly named!!!

For the design document kindly refer to section "Design Documentation" for details.

DESIGN DOCUMENTATION

For the design document provide write-up for the following sections:

1. Design
    a. Describe your thoughts and overall approach, including the breakdown of your logic
    b. Describe the different python objects that you use to develop your program
    c. Describe the program flow, that is, the structure of your program
    d. Describe the logic used to generate the randomized puzzle, both 8 and 15's.
2. Functions
    a. Describe usage of all your newly defined functions, including details of parameter(s)
3. Output
    a. Show samples of output from your program

TIPS & HINTS

- Beware of variable scope as you might keep a few variables as global such as puzzle
- Refer to python website for program styles and naming convention (PEP 8)
- Validate all inputs - if incorrect input is detected then display a friendly response and prompt the input again.

SAMPLE OUTPUT

Welcome to 8-puzzle game, …..

Enter the four letters used for left, right, up and down directions > l r u d

Enter "1" for 8-puzzle, "2" for 15-puzzle or "q" to end the game > 1

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

Enter your move (left-l, up-u) > l

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

Enter your move (left-l, right-r, up-u) > u

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

Enter your move (left-l, right-r, up-u, down-d) > l

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

Enter your move (right-r, up-u, down-d) > u

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

Congratulations! You solved the puzzle in 4 moves!

Enter "1" for 8-puzzle, "2" for 15-puzzle or "q" to end the game> 1

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

Enter your move (left-l, right-r, up-u, down-d) >

.
.
.

MARKING CRITERIA

- Coding Styles – overall program structure including layout, comments, white spaces, naming convention, variables, indentation, functions with appropriate parameters and return.
- Design Documentation
- Program Correctness – whether or the program works 100% as per Scope.
- User Interaction – how informative and accurate information is exchanged between your program and the player.
- Readability counts – programs that are well structured and easy-to-follow using functions to breakdown complex problems into smaller cleaner generalized functions are preferred over a function embracing a complex logic with nested conditions and sub-functions! In other words, a design with clean architecture with high readability is the predilection for the course objectives over efficiency.
- KISS approach – Keep It Simple and Straightforward.
- Balance approach – you are not required to come up a very optimized solution. However, take a balance between readability and efficiency with good use of program constructs.

ITEMS	PERCENTAGE	REMARKS
DESIGN DOC	10%-15%	
CODING STYLES	20%-25%	0% IF PROGRAM DOESN'T RUN
USER INTERFACE	15%-20%	0% IF PROGRAM DOESN'T RUN
FUNCTIONALITY	>40%	REFER TO SCOPE

DUE DATE

March 22nd, 2020, 11:59:59PM