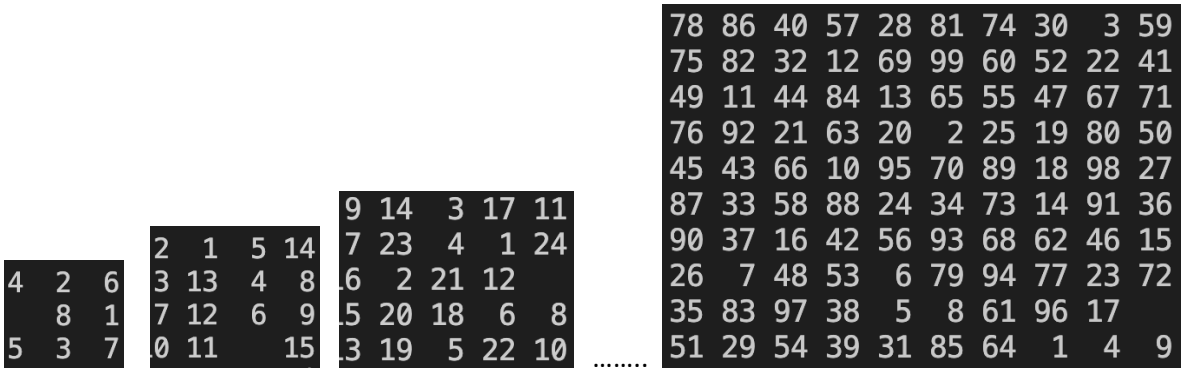
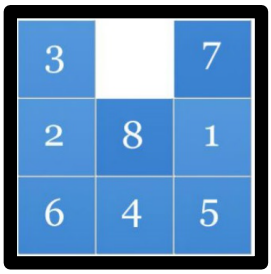
# Sliding Puzzle – 2021



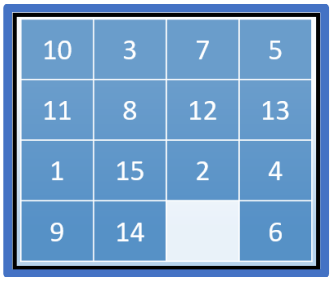

# OVERVIEW

In this assignment, you are going to design and develop an interactive sliding puzzle game where users can choose any dimensions up to 10x10, minimum dimension is 3x3. For a 3x3 puzzle, it has a square-framed board consisting of 8 square tiles, numbered 1 to 8, initially placed in random order as shown in the following figure.



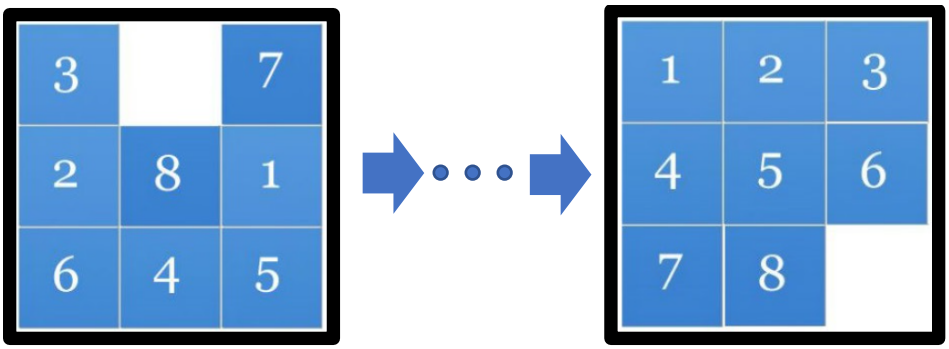

3x3 puzzle

A 4x4 puzzle has 15 numbered square tiles, from 1 to 15, as shown below.




4x4 puzzle

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 3x3 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 for the desired dimension of the puzzle, minimum 3, up to and including 10.
3. 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.
4. After the last prompt, generate a randomized, SOLVABLE puzzle accordingly; have it displayed on the screen using simple ASCII characters.
5. Game begins by repeatedly prompting the 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) >
6. After each move, show the updated puzzle on the screen and prompt further direction if needed.
7. 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.
8. Track total number of moves made for each game and have it displayed as the puzzle is solved.
9. Validate all inputs.

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. Overview
    b. Data Model
        i. describe core data objects called Data Model (such as list, string, dictionary and so on) that you used to develop your program for representing the puzzle and tracking the sliding position.
    c. Program Structure (your thoughts and overall approach)
        i. describe the breakdown of your logic into various functions and how these functions are organized (basically the structure of your program)
    d. Processing Logic
        i. Describe the main processing logic.
        ii. Describe your technique you developed to generate the randomized puzzle.
2. Function Specification
    a. Describe usage of all your own defined functions, including details of parameter(s) and output if any.
3. Output
    a. Show samples of output from your program

    Note: See Appendix for a template of design doc.

# 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 Kinley's puzzle game, …..

Enter the desired dimension of the puzzle > 3

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

```
   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 'n' to start a new game or enter 'q' to end the game > n

```
 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 not 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 with 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 21st, 2021, 11:59:59PM

# APPENDIX - TEMPLATE

## Design Doc

### OVERVIEW

- Write one or two paragraphs to describe in high-level what your program does.

### DATA MODEL

- Describe the type(s) of data that you used to model your core objects in your program. In this case, it will be the game puzzle as well as the direction keys, plus any other objects that you think of essential to mention.

### PROGRAM STRUCTURE

- Describe the structure of your program, specifically how you organized your thoughts in terms of functions and how these functions are organized. In general, you might breakdown your logic into many functions organized by functional components, each of which performs specific role. For an example, you might have one component to create the initial puzzle, one component to handle the update of the data model for the puzzle, one component to display the puzzle, one component to handle the interaction with the users and so on.
- For each component, describe the role and list out the function(s) in high level, details to be included in section Functional Specifications.

### PROCESSING LOGIC (SPECIFIC)

- Main processing logic - describe how you piece together various components and data model to implement your program.
- Initial Puzzle - describe the technique used to generate the initial, randomized puzzle.

### FUNCTIONAL SPEC

- Describe usage of all of your own defined functions including an overview description, detailed description of parameters, as well as output if any.

### SAMPLE OUTPUT

- Include a few samples of outputs from your program.