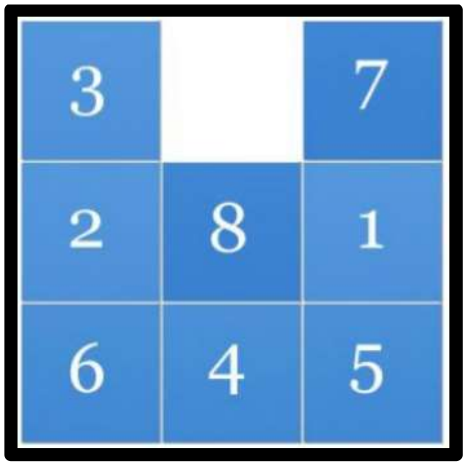
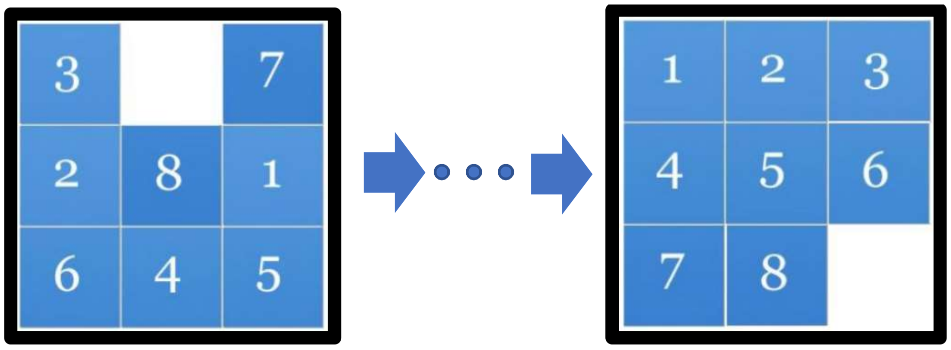
8-Puzzle game



OVERVIEW

In this assignment, you are going to design and develop an interactive 8-puzzle game. It has a square-framed board consisting of 8 numbered square tiles initially placed in random order. 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 in sequential order by repeatedly making sliding moves.



INITIAL FINAL

SCOPE

1. At the start of the game, generate a randomized, SOLVABLE 8-puzzle, then have it displayed on the screen using simple ASCII characters.
2. Prompt player the sliding direction (left, right, up or down)
3. Display the updated 8-puzzle resulting from the move above, and prompt further direction if needed
4. Inform user when the puzzle is solved (i.e. the numbered tiles are in sequential order); then prompt user to continue or end the program.
5. Track total number of moves 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 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, or FE 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. Program structure and flow
    a. Describe the main process in turns of the overall program flow, that is, your solution to the problem presented by this assignment.
2. Python objects (global variables)
    a. Usage of core python objects (purposes)
3. Functions
    a. Describe usage of all your newly defined functions, including details of parameter(s)
4. 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, ………

Press any key to begin >

             1   3
         4   2   5
         7   8   6

Input sliding direction (left, up) > l

         1       3
         4   2   5
         7   8   6

Input sliding direction (left, right, up) > u

         1   2   3
         4       5
         7   8   6

Input sliding direction (left, right, up, down) > l

         1   2   3
         4   5
         7   8   6

Input sliding direction (right, up, down) > u

         1   2   3
         4   5   6
         7   8

Congratulations!  You solved the puzzle in 4 moves!

Do you want to start a new game (Y/N)? Y

          8  1  3
          4     2
          7  6  5

Input sliding direction (left, right, up, down) >
                    .
                    .
                    .
```

MARKING CRITERIA

- Coding Styles – layout, comments, white spaces, naming convention, variables, indentation.
- Documentation
- Program Correctness – logic, program structure, functions with appropriate parameters
- 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.

CHALLENGE 1 (NOT REQUIRED)

DO NOT INCLUDE THIS SOLUTION IN YOUR CURRENT ASSIGNMENT, HAVE IT CODED SEPERATELY AND PRESENTED TO ME IN PERSON!!!

Implement a solution to "programmatically" solve the puzzle. In this case, your program no longer prompts user for anything; instead it will display the initial randomized puzzle and then sequentially display the updated puzzle for each sliding move your program decided to make, and finally the solved puzzle.

No additional mark will be credited to your assignment, this is solely for your self-interest purposes.

CHALLENGE 2 (NOT REQUIRED)

DO NOT INCLUDE THIS SOLUTION IN YOUR CURRENT ASSIGNMENT, HAVE IT CODED SEPERATELY AND PRESENTED TO ME IN PERSON!!!

Another challenge is to determine if a puzzle is invariant, that is, not solvable. In this case, your program must generate a 8-puzzle for this purpose.

No additional mark will be credited to your assignment, this is solely for your self-interest purposes.

CHALLENGE 3 (NOT REQUIRED)

DO NOT INCLUDE THIS SOLUTION IN YOUR CURRENT ASSIGNMENT, HAVE IT CODED SEPERATELY AND PRESENTED TO ME IN PERSON!!!

Generalize your solution to solve for larger puzzle such as 15-puzzle and 24-puzzle, or even larger ones.

No additional mark will be credited to your assignment, this is solely for your self-interest purposes.

DUE DATE

March 10th, 2019, 11:59:59PM