9 WAY PUZZLE

# Documentation

van Wyk, JJ

25950665

## Table of content

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Part of ... | Page |
| Introduction |  | Assignment 1 | p.2 |
| Explanation |  | Assignment 1 | p.3-p.6 |
| What recourses I used |  | Assignment 1 | p.6-p.8 |
| A summary of how the code works |  | Assignment 1 | p.9 |
| Implementing Search functionality |  | Assignment 2 | p.10-p.16 |
| Implementing Search functionality |  | Assignment 2 | p.16 |
| Experimental analysis |  | Assignment 2 | p.16 |
| References |  | Assignment 1 & 2 | p.17-p.18 |

# Introduction

For this assignment I have created a working 9-puzzle/8-puzzle in Python. The game rules are as follows:

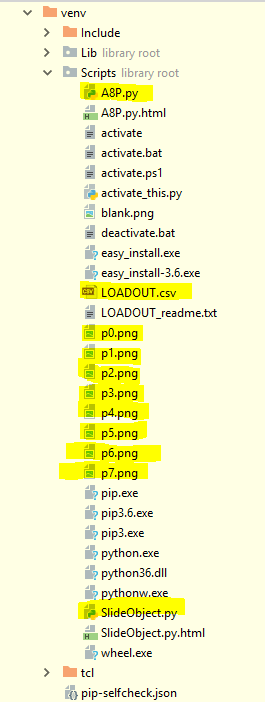
* 3x3 random numbered squares placed on a board with 1 square missing.
* Try to align the number 1-8 by only moving the squares into the empty square.
* Only a square that is next to an empty square may be moved into the empty slot.
* Squares can also form an image instead of numbers.
* Number of moves are counted and display at the end.

The lectures 'BoilerPlate.py' file was used as a base for my project. Pycharm was used to develop my project.

Pygame library is what is used to create my projects GUI.

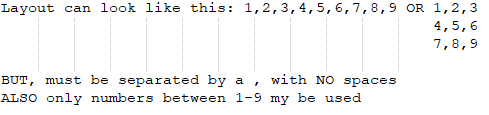
# Explanation

Pycharm is needed to run the project.

Under \Assi\_8P\venv\Scripts\ run A8P.py

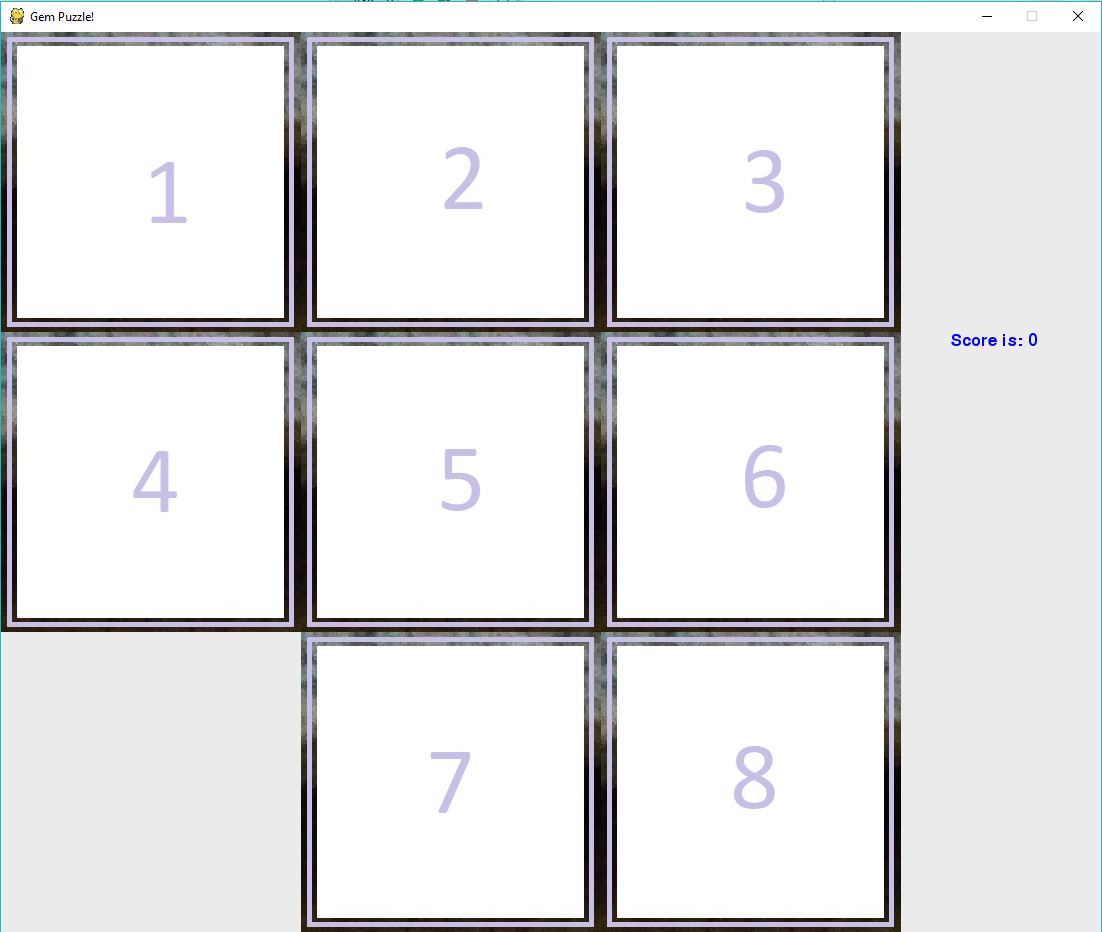
Open 'A8P.py', 'SlideObject.py' and 'LOADOUT.csv'

Download CSV Plugin

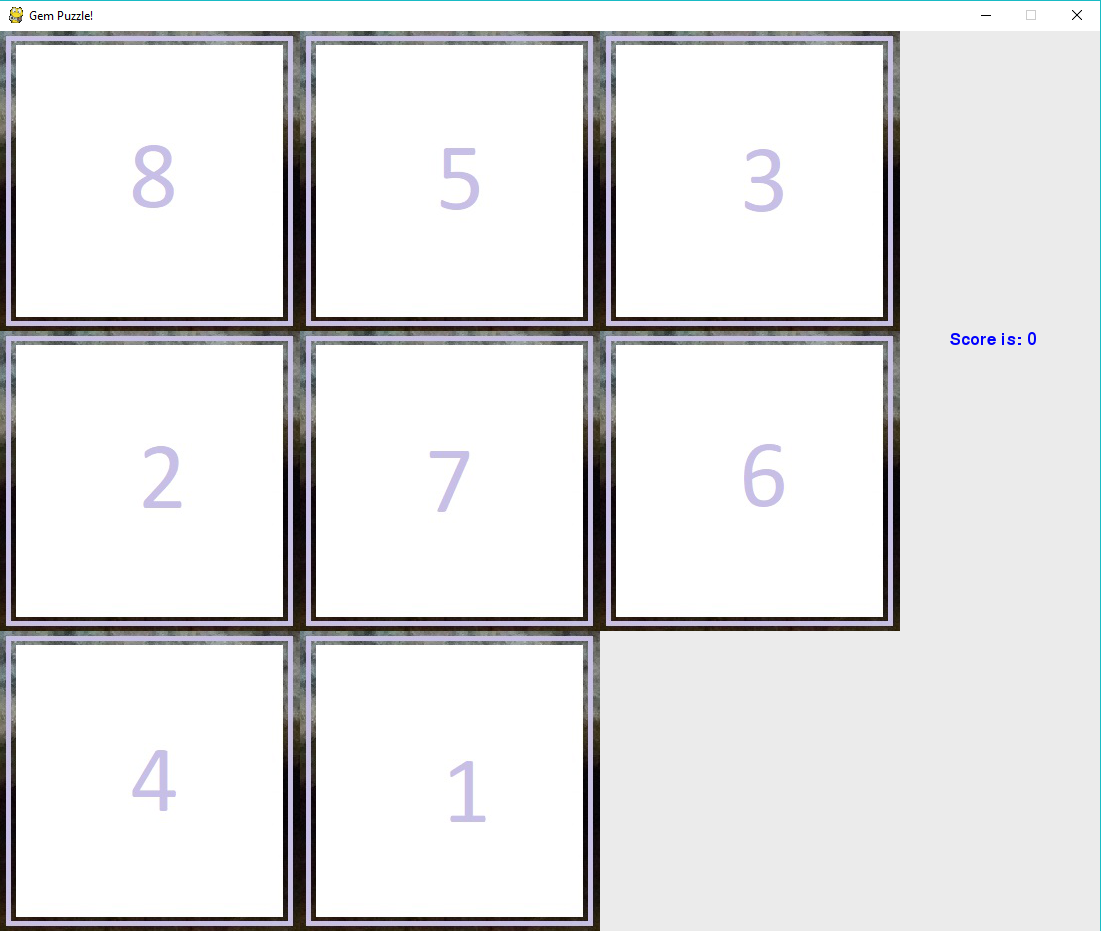
LOADOUT.csv layout: 

(9 is the empty space)

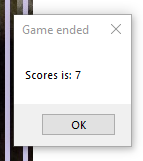
On running 'A8P.py' screen should look like this if layout given was 1,2,3,4,5,6,9,7,8:



If 'LAYOUT.csv' don't exist or is incomplete, a set of 9 random numbers will be made in the place of the layout:

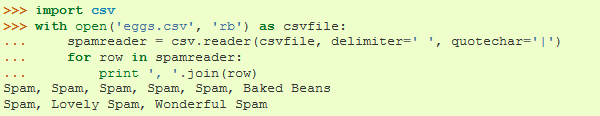


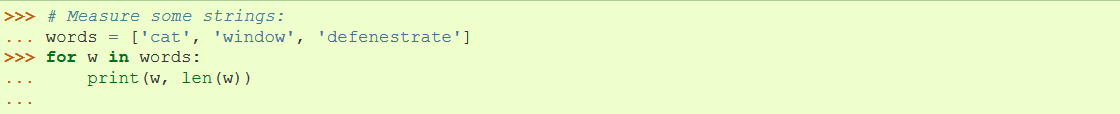
In this example only square 1 and 6 will move into the empty space.

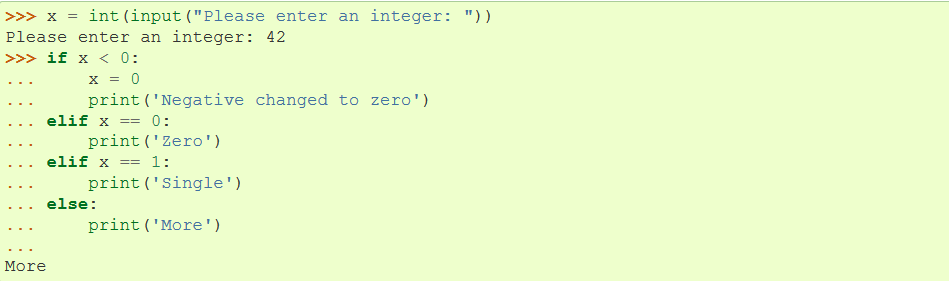
If you exit or complete the game, your score will be shown:

What recourses I used

I used Python Foundations Forums to learn how to use .csv file, if-statements and for-loops (with objects and range()):









And also used stackoverflow forums for learning:

* Converting string to int and float, and in reverse
* How to get a number of elements in a list, len()
* Code for how to check if a file exists:

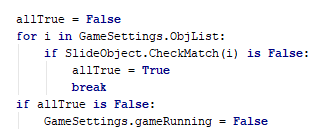


* Creating simple popup dialogs:



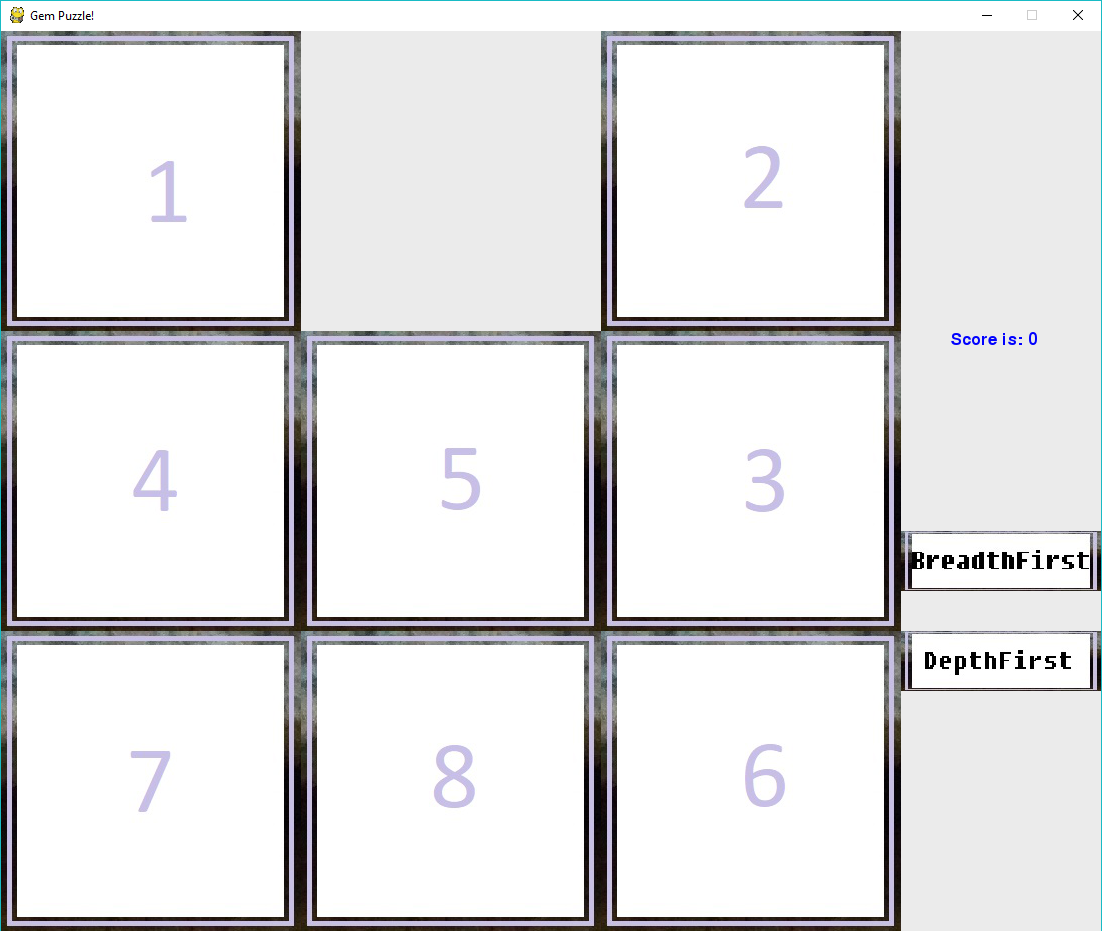
I also learned the basics of Python by watching 'thenewboston' tutorial playlist from 1 – 18

# A summary of how the code works

1. First thing that happens (with the code I added) is that a list of positions for the slides are made from the .csv file or randomly if the .csv file doesn't exist.
2. That list is use to place images in their corresponding positions.
3. "Hitboxes" are made for all these positions (Info from 2 and 3 is stored in a list of SlideObjects)
4. If the click event is triggered:
   1. Checks all the possible spaces that slides around the empty space can move into
   2. Checks if the position you click on is in the "Hitbox" of one of the possible spaces
   3. Switches that slides position with the empty space's position
5. After all the moving is done:
   1. The playing field gets redrawn
   2. Checks if all of the pictures align with the correct positions:
      1. Then ends game
      2. 

# Implementing Search functionality

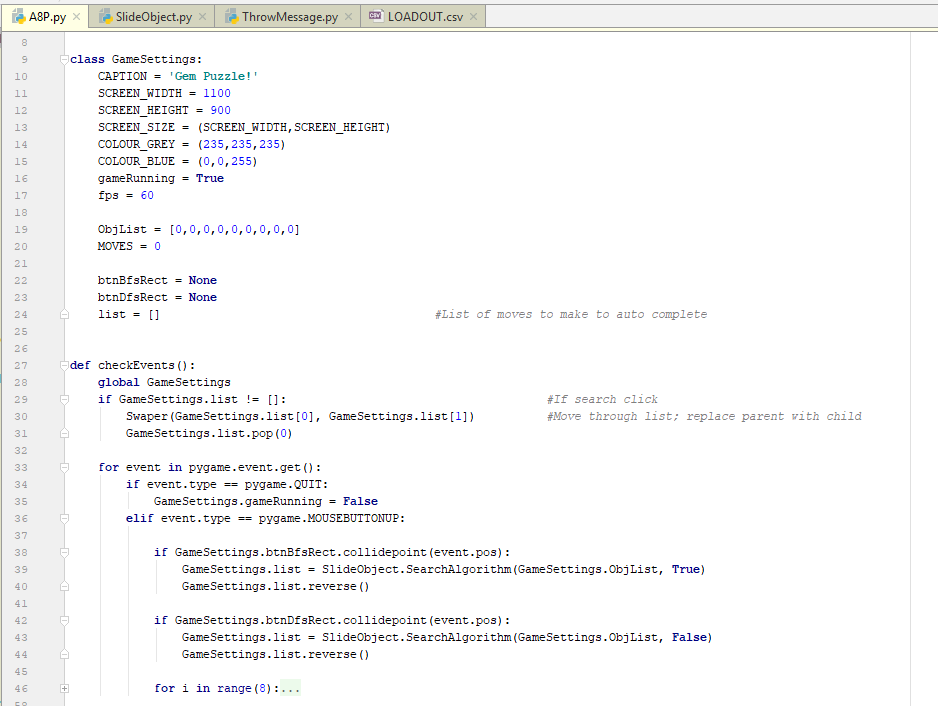
First off lets me show you what and where the triggers are that start each search method to autocomplete the remainder of the puzzle:



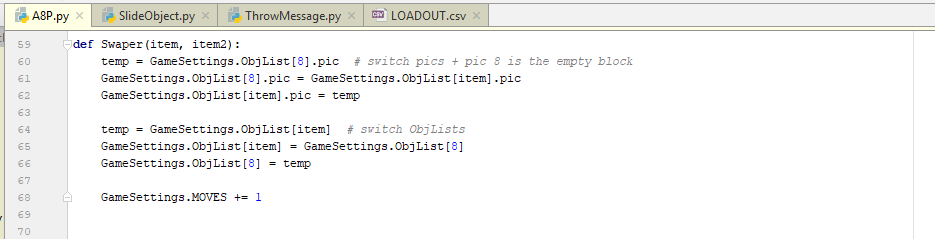
This board-state will be used as an example from here on out

Clicking on one of the "buttons" will start each search respectively

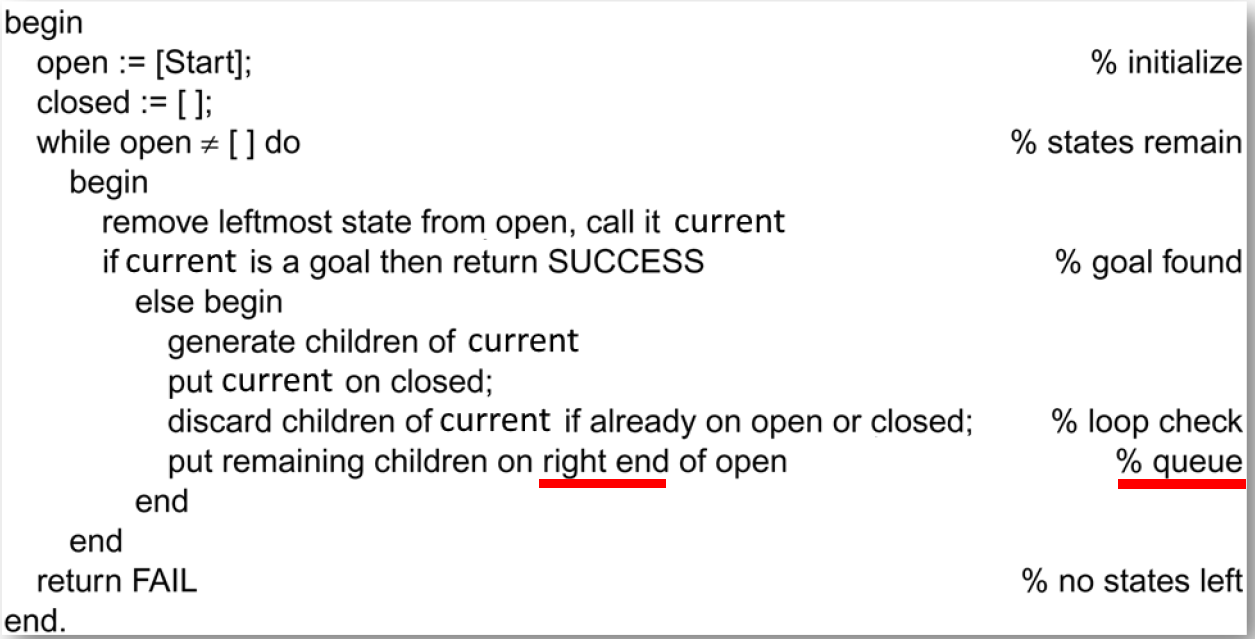
Code trigger:

Simply put, if the button is clicked, a list will be made (of the given search type) using the current position of the slides. The is then reversed. After this nothing will happen until checkEvent() runs again after a tick, then before checking the events an if-statement will run (since list[] isn't empty anymore). Using the list[] it will take the moves 2 at a time and swap the pieces on the board, removing the old movement from the list each tick/run until list is empty again

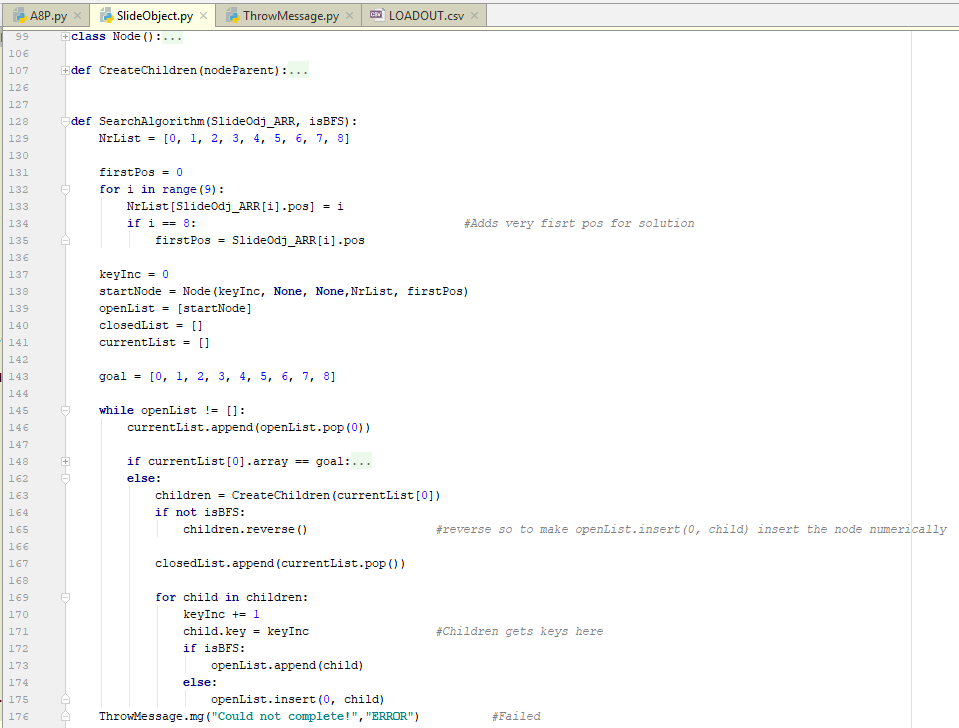
Swap method:



Now finally how I implemented the search methods:



Using the pseudo code given to us as a guide to what steps to follow, I did the following code:

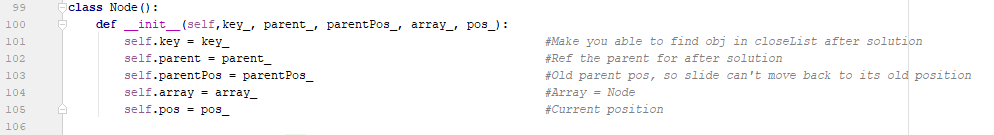


(This method works for both Breadth & Depth first algorithms)

From line 129 – 135: It simply creates an array of all the current positions of the slides (also saves the start position of the empty space, slide 8), by getting the 'pos' variable from each 'SlideObject' in the SlideOdj\_ARR[] list, ex. [0,8,1,3,4,2,6,7,5] for the board in the first image (numbers go from 0-8, not 1-9)

Now for the important part, lines 137 – 176:

Line 137 – 145: As stated in the pseudo code 3 lists are made (openList, closedList and currentList), where openList is initiated with the root/starting node (also setting up from key references at line 137). In this case the node is a class named 'Node', which look and does the following:

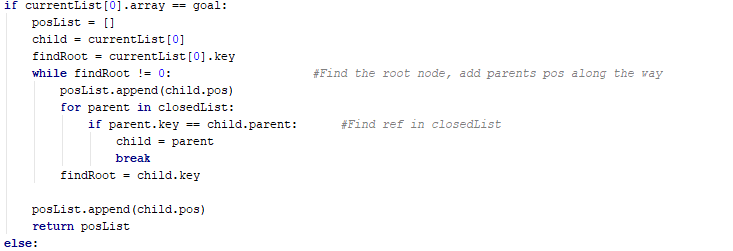


In line 138 startNode would look like this:

Node (0, None, None, [0,8,1,3,4,2,6,7,5], 1)

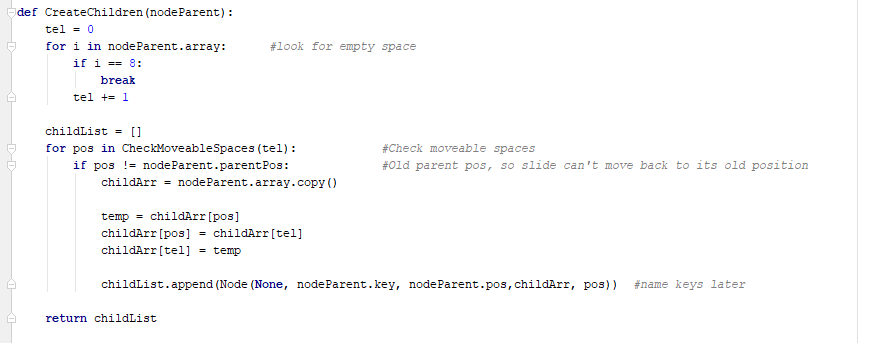
Line 145 & 176: Loop while list isn't empty and if list is empty (goal not found in all of the iterations) throw an error message to end game

Line 148: Check if currentList is [0,1,2,3,4,5,6,7,8], if it is do this:



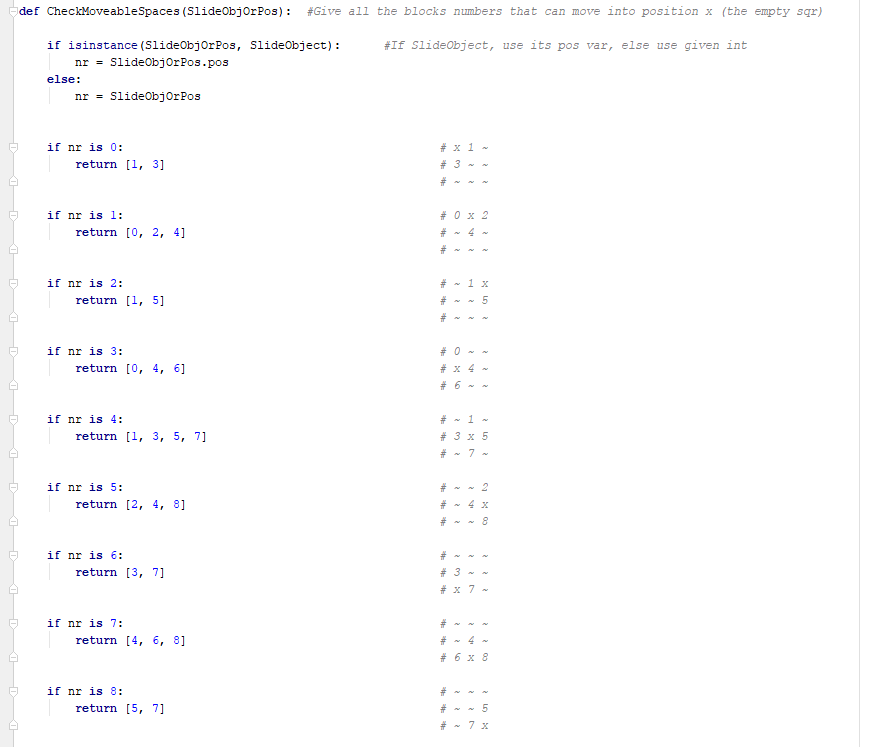
Looks at parent (reference of parent in child key) in closedList and adds the parents pos/position to a list. Examples result will be [8,5,2,1], which means the slide on position 1 swaps with 2, 2 with 5 and 5 with 8.

Line 163: Generate children of currentList, put it into a list named children:



This is where the children nodes are created and also where it starts getting complicated …

First, it should find the empty slide (aka. 8) in the parent node (currentList's item), put it into 'tel'. Then loop for every position that can move into the empty space (CheckMoveableSpaces(int) ).



As long as it isn't a slide that has already been in the space (parent-node's parent's position = the previous slide that moved), *this is to prevent Depth first algorithm to not move between only the first 2 moves for the rest of infinity.* Lastly, the nodes are assigned to a list. Key are assigned later.

Line 164 – 165: This if-statement will run if the depth first algorithm is being used. It is to make it search for the LEFT side, as this is convention

Line 167: currentList's item is appended to closedList

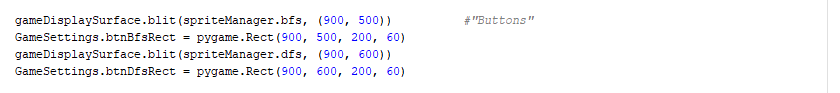
I didn't do the step in the pseudo code that discards the children of currentList if it is already on openList or closedList, check loop. Was not sure where this would fit in or if it was even necessary.

Line 169 – 175: This adds the children node to the openList aswell as assigns unique keys to the children. The way it adds the child-nodes to openList depends on the search algorithm selected. If you are using breadth first (isBFS = True), then child-node is simply appended to the right side of the openList. If you are using depth first (isBFS = False), then each item (which is reversed) will be added to the first position in the openList.

# Decisions made that changed the game

Very little about how you play the game has changed since assignment 1, besides the "buttons" added to the side that will auto-complete your current broad state using the 2 search methods, no matter how many moves you made beforehand. Score is still display at the end of the game even if it was auto-completed.

Buttons were created as such:



# Experimental analysis

I believe that the Breath first method for the puzzle is far superior to Depth first method.

I believe the reason for this is that the Depth first algorithm (when starting with the left side of the decision tree, as accustom) is that the depth of a single branch in the puzzle has so many levels before it reaches the deepest level, that even completing one branch can take an extremely long time.

Breadth first algorithm works through every level first, which means it will check every move made in a single turn over mutable decisions before moving, before moving to the next turns moves and decisions.

But, breath first will work faster on puzzles where there aren't too many turns needed to complete a game, while depth first will most likely be faster then there are a lot of turns before a game is completed.

## References

Docs.python.org. (2018). *4. More Control Flow Tools — Python 3.5.5 documentation*. [online] Available at: <https://docs.python.org/3.5/tutorial/controlflow.html> [Accessed 23 Mar. 2018].

Docs.python.org. (2018). *13.1. csv — CSV File Reading and Writing — Python 2.7.14 documentation*. [online] Available at: <https://docs.python.org/2/library/csv.html> [Accessed 24 Mar. 2018].

Docs.python.org. (n.d.). *9.6. random — Generate pseudo-random numbers — Python 2.7.14 documentation*. [online] Available at: <https://docs.python.org/2/library/random.html> [Accessed 23 Mar. 2018].

Stackoverflow.com. (2018). *Converting integer to string in Python?*. [online] Available at: <https://stackoverflow.com/questions/961632/converting-integer-to-string-in-python> [Accessed 23 Mar. 2018].

Stackoverflow.com. (2018). *How can I create a simple message box in Python?*. [online] Available at: <https://stackoverflow.com/questions/2963263/how-can-i-create-a-simple-message-box-in-python> [Accessed 25 Mar. 2018].

Stackoverflow.com. (2018). *How do I parse a string to a float or int in Python?*. [online] Available at: <https://stackoverflow.com/questions/379906/how-do-i-parse-a-string-to-a-float-or-int-in-python> [Accessed 23 Mar. 2018].

Stackoverflow.com. (2018). *How to check whether a file exists?*. [online] Available at: <https://stackoverflow.com/questions/82831/how-to-check-whether-a-file-exists> [Accessed 24 Mar. 2018].

Stackoverflow.com. (2018). *How to get the number of elements in a list in Python?*. [online] Available at: <https://stackoverflow.com/questions/1712227/how-to-get-the-number-of-elements-in-a-list-in-python> [Accessed 23 Mar. 2018].

YouTube. (2018). *Python 3.4 Programming Tutorials - YouTube*. [online] Available at: <https://www.youtube.com/playlist?list=PL6gx4Cwl9DGAcbMi1sH6oAMk4JHw91mC_> [Accessed 22 Mar. 2018].

New refs:

Stackoverflow.com. (2018). *creating rect Buttons with Pygame*. [online] Available at: https://stackoverflow.com/questions/40336960/creating-rect-buttons-with-pygame [Accessed 05 May. 2018].

Stackoverflow.com. (2018). *What's the canonical way to check for type in Python?*. [online] Available at: https://stackoverflow.com/questions/152580/whats-the-canonical-way-to-check-for-type-in-python [Accessed 05 May. 2018].

Stackoverflow.com. (2018). *Data Structures*. [online] Available at: https://docs.python.org/3/tutorial/datastructures.html[Accessed 05 May. 2018].

Stackoverflow.com. (2018). *How do I copy an object in Python?*. [online] Available at: https://docs.python.org/3/faq/programming.html#how-do-i-copy-an-object-in-python[Accessed 05 May. 2018].

Stackoverflow.com. (2018). *Errors and Exceptions* [online] Available at: https://docs.python.org/2/tutorial/errors.html[Accessed 05 May. 2018].