(1) Summary:

As will be stated in the conclusion, my project features Markov-Chain logic, a topic that I learned in my Probability course (Math 218). In this regard, I treated each state of my game as a temporary object using “self.\_\_\_” functions, and created sub-functions accordingly. This temporary object has state based dependence only upon the previous state, and is continuously changing. Using this logic, and after some research into pygame functions and mechanics, I was able to recreate a Sliding Puzzle from scratch.

(2) Background:

The game is simple, and utilizes different tiles that can switch if they are adjacent to each other. The goal of the game is to put the tiles in order from smallest to largest, top left to bottom right, and to put the empty tile in the bottom right. The tiles are numbered from 1 to 15. A tile can only be switched with the empty tile.

To accomplish this task, I created 10 functions in total, all with different purposes. I defined eight sub-functions within one of the main functions of the program, \_\_in\_it\_\_. The set\_as\_blank and getblank functions are used to set one of the tiles as an empty tile, an integral part of the design of the game. This allows for there to be a space in which each desired tile is moved to. The adjacent function defines legal moves in the game as those which are tiles adjacent to the opentile. The in\_grid function defines whether or not a given tile is within the array, and only allows moves to be made within the grid. The switch function defines the method of how the desired tile and the open\_tile. The random function makes a random choice between the tiles adjacent to the open\_tile depending on the position of the open\_tile. The update function defines the action of how to switch with a desired tile if a mouse is clicked. The draw function creates the graphics of the tiles for each tile. The events function defines the action of how to switch with a desired tile if a given w,a,s, or d key is pressed. Everything in the main function defines all of the continuously running parts of the program. All of the constants are listed above, such as the amount of tiles, position of tiles, margin size, screen size, text size, font, color of tiles, and so on. Also listed there is the for loop that randomizes the initial state of the game.

(3) Motivation:

I chose this particular project, because I knew the Sliding Puzzle was heavily dependent upon labeling, and the concept of arrays. I also thought that it would be interesting and challenging to attempt to create a “flash-like” game outside of Java. I also found the idea of creating graphics from Python to be intriguing, and the Sliding Puzzle was a game that I grew up with, so I was familiar with the mechanics of the game. I was initially debating which game to choose, between Towers of Hanoi, Connect Four, and the Sliding Puzzle, however after learning of another group doing Connect Four, and after realizing that except for the graphics, Towers of Hanoi was not as challenging, I decided to go with the Sliding Puzzle.

Contributions:

* Considering that I was a 1-person group for the project, I contributed all of the components for it, such as:
  + The Report
  + The Code
  + The Powerpoint presentation (both proposal and final)

(4) Methods, Packages, etc.:

My main method was to draw out the logic of the puzzle on paper, creating a diagram of how the array must be made, and from what variable (i.e. how to define length of tiles, how to define text on tiles using a for loop, etc.).

I quickly realized that the graphics would prove to be a challenge, and thus had to abandon a way of thinking of the world object, i.e. the array of the game itself, that was consistent with what we learned in class. Instead of thinking of it as a consistent object, I had to think of it as one which was always changing, which is why I began to look into the “self” functions and operations. After reading up on this and the functions associated with it in pygame, it became easier to refer to the game as a continuously changing Markov-Chain based object (i.e. an object whose current state is not dependent on any other state except the previous one). In this respect, the majority of my logic was now based on “self.\_\_\_\_\_” functions, which also comprise the majority of my code. While these functions are not consistent with what I learned in class, many of the less obvious parts of my code are, such as for loops, while loops, array set-up, array labeling, switching definition/order of variables, using equalities/inequalities, indexing, dictionaries, string functions, comparisons, etc.

As far as packages, I utilized the pygame package to access certain functions vital in the production of the Slide Puzzle Game, many of which are listed at this link:

<https://www.pygame.org/docs/tut/newbieguide.html>

I also utilized the os package to access the operating system under Python to display the screen containing the array of the game. I used the random package to randomize the game, and to let the player randomize further if necessary by pressing space, and I used the sys package to access system-specific parameters and functions, many of which are listed at this link: <https://docs.python.org/2/library/sys.html#:~:targetText=sys%20%E2%80%94%20System%2Dspecific%20parameters%20and%20functions,It%20is%20always%20available.&targetText=If%20no%20script%20name%20was,0%5D%20is%20the%20empty%20string.>

(5) Conclusion:

Overall, I believe the project went very well. There are a few things I would have done to add to the project given more time, however I am proud of the progress I made. I learned about “self.\_\_\_” functions in this project, which I know will come in handy in the future whenever I need to refer to temporary objects. While working with pygame seems to be a niche of Python as a whole, working with randomization and probability, and working on creating graphics was an interesting challenge. I now know how to, and why to import certain packages, which will also be handy for the future.

If I had more time, I would have implemented a method for the player to choose the dimensions of the game. I would have also had the tiles be displayed as pieces of an image, rather than all the same color. In addition, I would have implemented a timer that the player would be able to choose themselves.

(6) References:

1. <https://www.geeksforgeeks.org/self-in-python-class/#:~:targetText=self%20represents%20the%20instance%20of,attributes%20with%20the%20given%20arguments.&targetText=self%20is%20parameter%20in%20function,name%20in%20place%20of%20it.>
2. <https://pythontips.com/2013/08/07/the-self-variable-in-python-explained/>
3. <https://www.w3schools.com/python/python_classes.asp>
4. <https://www.pygame.org/docs/tut/newbieguide.html>
5. <https://inventwithpython.com/pygame/chapter4.html>
6. <https://docs.python.org/2/library/sys.html#:~:targetText=sys%20%E2%80%94%20System%2Dspecific%20parameters%20and%20functions,It%20is%20always%20available.&targetText=If%20no%20script%20name%20was,0%5D%20is%20the%20empty%20string.>

**Note: No table is needed to show member contributions, as I was the only member of my group. The report took around 4-5 hours, while the code took around 14 hours to complete with logic and critical thinking.**