**A Project on**

**8-Puzzle Game using python**

**Prepared by-**

M.Ramkoushik

**ABSTRACT**

Python is an ideal language for building an 8-puzzle game due to its simplicity, flexibility, and extensive libraries, making it an excellent choice for developers of all skill levels. With Python, implementing the game’s logic is straightforward, as its syntax and nature allow for easy representation of the game’s rules and constraints, such as handling valid moves, checking for the puzzle's solution, and managing the state of the game board. Additionally, Python’s built-in data structures like lists and dictionaries can be used to represent the 3x3 grid, making it easy to manipulate and access the game state. The random module can be used to generate random initial states for the puzzle, ensuring that each game is unique and challenging. Furthermore, Python’s extensive range of GUI libraries, including Tkinter, PyQt, and Pygame, can be used to create a user-friendly interface for the game, allowing players to interact with the game easily. Moreover, Python’s simplicity and flexibility make it easy to implement various solving algorithms, such as the A\* algorithm, breadth-first search, or depth-first search, enabling developers to experiment with different approaches and optimize game performance. Overall, Python's ease of use, flexibility, and extensive libraries make it an excellent choice for building an 8-puzzle game, allowing developers to focus on creating an engaging and challenging experience for players.

**INDEX**

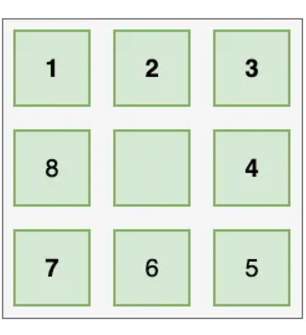
|  |  |  |
| --- | --- | --- |
| **SI.NO** | **TITLE** | **PAGE NO** |
| 1  2  3  4  5 | Rules of game  Writing the 8-Puzzle code in Python  Understanding the code  Run the code using python Compiler  Conclusion for 8-Puzzle game | 4  5  6  7  8 |

**Step-1:- Rules of the game**

The 8-puzzle game is a sliding puzzle that involves a 3x3 grid with 8 numbered tiles (from 1 to 8) and one blank space. The goal of the game is to rearrange the tiles into a specific configuration, usually with the numbers in ascending order (1 to 8) with the blank space at the bottom-right corner. The game is played by sliding tiles into the blank space, and the movement of tiles is restricted to adjacent positions.

* The 8-puzzle consists of 8 numbered tiles and 1 blank space on a 3x3 grid.
* The goal is to arrange the tiles in numerical order, with the blank space at the bottom-right corner.
* Tiles can only move into the blank space, and they must be adjacent (horizontally or vertically).
* The game is won when the puzzle reaches the goal configuration.
* Not all configurations are solvable, and a puzzle is solvable if the number of inversions is even.

These rules make the 8-puzzle a challenging and engaging game, requiring spatial reasoning and problem-solving skills. Players must strategically move tiles into the blank space to reach the goal configuration, with each move demanding careful planning. The random starting setup and the constraint of sliding tiles into the blank space keep the game exciting and mentally stimulating.



**Step-2:-Writing the 8-Puzzle code in python**

To write the code for the 8-puzzle game, we can use Python as well. Simple 8-puzzle game where the player can move tiles into the blank space by entering commands.

Python code:-

import numpy as np

class Puzzle8:

def \_\_init\_\_(self, board=None):

if board is None:

self.board = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 0]])

else:

self.board = np.array(board)

self.empty\_pos = tuple(np.argwhere(self.board == 0)[0])

def display(self):

print(self.board)

def move(self, direction):

x, y = self.empty\_pos

if direction == 'up' and x > 0:

self.board[x, y], self.board[x - 1, y] = self.board[x - 1, y], self.board[x, y]

self.empty\_pos = (x - 1, y)

elif direction == 'down' and x < 2:

self.board[x, y], self.board[x + 1, y] = self.board[x + 1, y], self.board[x, y]

self.empty\_pos = (x + 1, y)

elif direction == 'left' and y > 0:

self.board[x, y], self.board[x, y - 1] = self.board[x, y - 1], self.board[x, y]

self.empty\_pos = (x, y - 1)

elif direction == 'right' and y < 2

self.board[x, y], self.board[x, y + 1] = self.board[x, y + 1], self.board[x, y]

self.empty\_pos = (x, y + 1)

else:

print("Invalid move!")

def is\_solved(self):

return np.array\_equal(self.board, np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 0]]))

def main():

puzzle = Puzzle8()

puzzle.display()

while not puzzle.is\_solved():

move = input("Enter your move (up, down, left, right): ").strip().lower()

puzzle.move(move)

puzzle.display()

print("Congratulations! You've solved the puzzle.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Step-3:- Understanding the code**

The **8-puzzle game code** in Python implements a sliding puzzle where the objective is to rearrange the tiles on a 3x3 grid to reach the solved configuration. The game is encapsulated in the Puzzle8 class, which contains methods to manage the puzzle board, make valid tile moves, and check if the puzzle is solved. The board is represented as a 3x3 NumPy array, with 8 numbered tiles (1-8) and one empty space (represented by 0). The init method initializes the puzzle, either with a scrambled state or the default solved configuration. The display method prints the current state of the grid, while the move method allows the player to move tiles in the four possible directions (up, down, left, right), provided the move is valid. The method checks if the puzzle is in the solved state, which is when the tiles are arranged in the order:

1 2 3

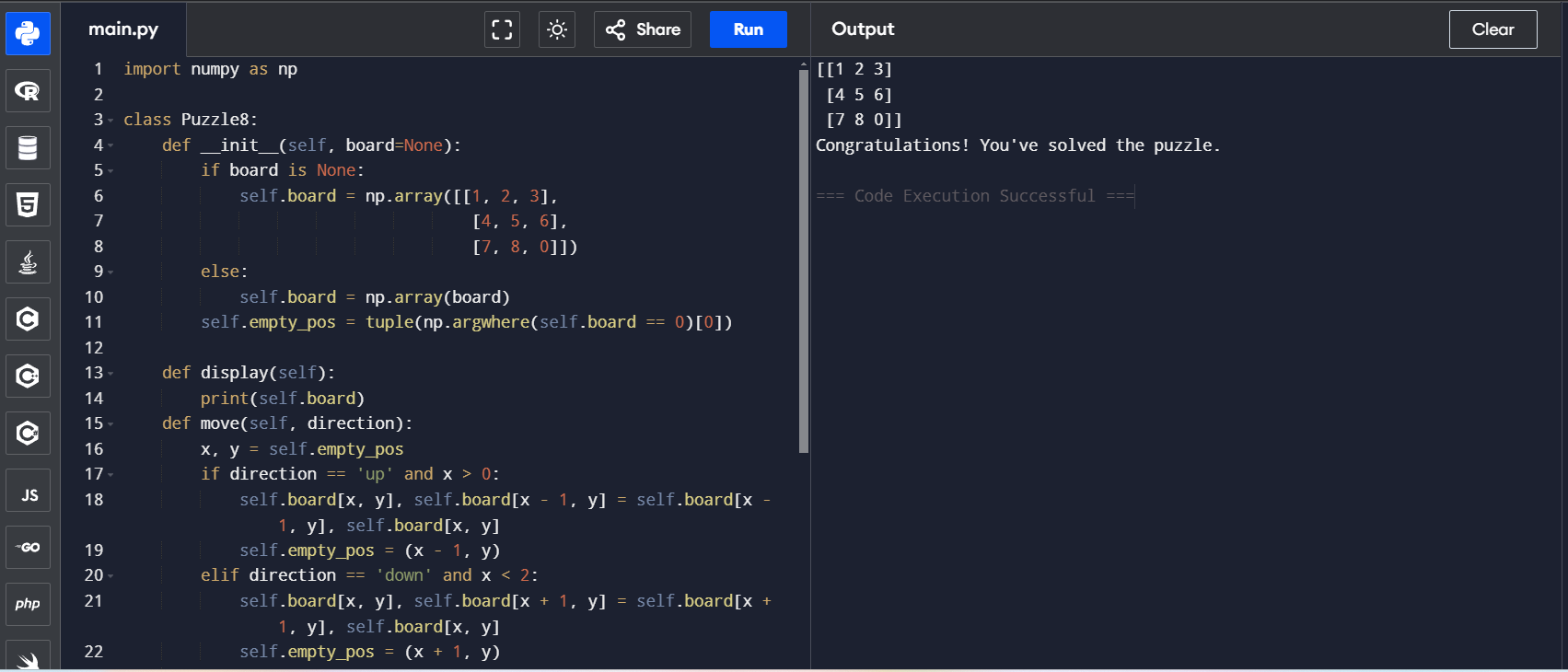
4 5 6

7 8 0

In the main function, a 8-Puzzle object is created, and the game begins by displaying the initial puzzle state. The player is prompted to enter a move (up, down, left, right), and the puzzle updates after each move. The game continues until the puzzle is solved, at which point a congratulatory message is displayed. The program ensures that only valid moves are allowed, and the game ends when the player successfully arranges the tiles in the correct order. This code offers a simple yet engaging 8-puzzle game, where the player can interactively solve the puzzle by moving tiles around the grid.

**Step 4: Run the code using Python Compiler**

**To run Python code for the 8-puzzle game, you can use an online Python compiler. Simply search for "Python compiler" online, select one, and run the code. The program will prompt you to enter moves like "up", "down", "left", or "right", and the puzzle will update until it's solved.**



**Step-5:- Conclusion for 8-Puzzle Game**

**In conclusion, the 8-puzzle game is an engaging and challenging puzzle that tests logical thinking and problem-solving skills. Through the Python implementation, we can easily manipulate the puzzle’s state by moving tiles around the grid, working towards the solved configuration. The game offers an interactive experience, where the player makes decisions that impact the puzzle’s progress. By using a simple yet effective algorithm to manage the moves and check for a solution, this implementation provides a great introduction to programming concepts like arrays, conditional statements, and user input handling. Overall, the 8-puzzle game is not only a fun way to practice programming, but also a rewarding puzzle-solving experience for players.**