8 puzzle solver code:

```
from simpleai.search import astar, SearchProblem


class PuzzleSolver(SearchProblem):

    def actions(self, cur_state):
        rows = string_to_list(cur_state)

        row_empty, col_empty = get_location(rows, 'e')
        actions = []

        if row_empty > 0:
            actions.append(rows[row_empty - 1][col_empty])
        if row_empty < 2:
            actions.append(rows[row_empty + 1][col_empty])
        if col_empty > 0:
            actions.append(rows[row_empty][col_empty - 1])
        if col_empty < 2:
            actions.append(rows[row_empty][col_empty + 1])

        return actions

    def result(self, state, action):
        rows = string_to_list(state)

        row_empty, col_empty = get_location(rows, 'e')
        row_new, col_new = get_location(rows, action)

        rows[row_empty][col_empty], rows[row_new][col_new] = (
            rows[row_new][col_new],
            rows[row_empty][col_empty]
        )

        return list_to_string(rows)

    def is_goal(self, state):
        return state == GOAL

    def heuristic(self, state):
        rows = string_to_list(state)
        distance = 0

        for number in '12345678e':
            row_new, col_new = get_location(rows, number)
            row_new_goal, col_new_goal = goal_positions[number]

            distance += abs(row_new - row_new_goal) + abs(col_new - col_new_goal)
```

```
        return distance


def list_to_string(input_list):
    return '\n'.join(['-'.join(x) for x in input_list])


def string_to_list(input_string):
    return [x.split('-') for x in input_string.split('\n')]


def get_location(rows, input_element):
    for i, row in enumerate(rows):
        for j, item in enumerate(row):
            if item == input_element:
                return i, j


GOAL = '''1-2-3
4-5-6
7-8-e'''

INITIAL = '''1-e-2
6-3-4
7-5-8'''


goal_positions = {}
rows_goal = string_to_list(GOAL)

for number in '12345678e':
    goal_positions[number] = get_location(rows_goal, number)


result = astar(PuzzleSolver(INITIAL))

for i, (action, state) in enumerate(result.path()):
    print()

    if action is None:
        print('Initial configuration')
    elif i == len(result.path()) - 1:
        print('After moving', action, 'into the empty space. Goal achieved!')
    else:
        print('After moving', action, 'into the empty space')

    print(state)
```

Output:

```
Initial configuration
1-e-2
6-3-4
7-5-8

After moving 2 into the empty space
1-2-e
6-3-4
7-5-8

After moving 4 into the empty space
1-2-4
6-3-e
7-5-8

After moving 3 into the empty space
1-2-4
6-e-3
7-5-8

After moving 6 into the empty space
1-2-4
e-6-3
7-5-8

After moving 1 into the empty space
e-2-4
1-6-3
7-5-8

After moving 2 into the empty space
2-e-4
1-6-3
7-5-8

After moving 4 into the empty space
2-4-e
```

```
After moving 3 into the empty space
2-4-3
1-6-e
7-5-8

After moving 6 into the empty space
2-4-3
1-e-6
7-5-8

After moving 4 into the empty space
2-e-3
1-4-6
7-5-8

After moving 2 into the empty space
e-2-3
1-4-6
7-5-8

After moving 1 into the empty space
1-2-3
e-4-6
7-5-8
```