BURUGU AJITH

MAIN PROJECT

DESCRIPTION:

Develop a Sudoku solver - a program that solves sudoku puzzles.

CODE:

```
def is_valid(board, row, col, num):
# Check if the number already exists in the row
    for i in range(9):
        if board[row][i] == num:
            return False

# Check if the number already exists in the column
    for i in range(9):
        if board[i][col] == num:
            return False

# Check if the number already exists in the 3x3 box
    start_row = 3 * (row // 3)
    start_col = 3 * (col // 3)
    for i in range(3):
        for j in range(3):
            if board[start_row + i][start_col + j] == num:
                return False

    return True
```

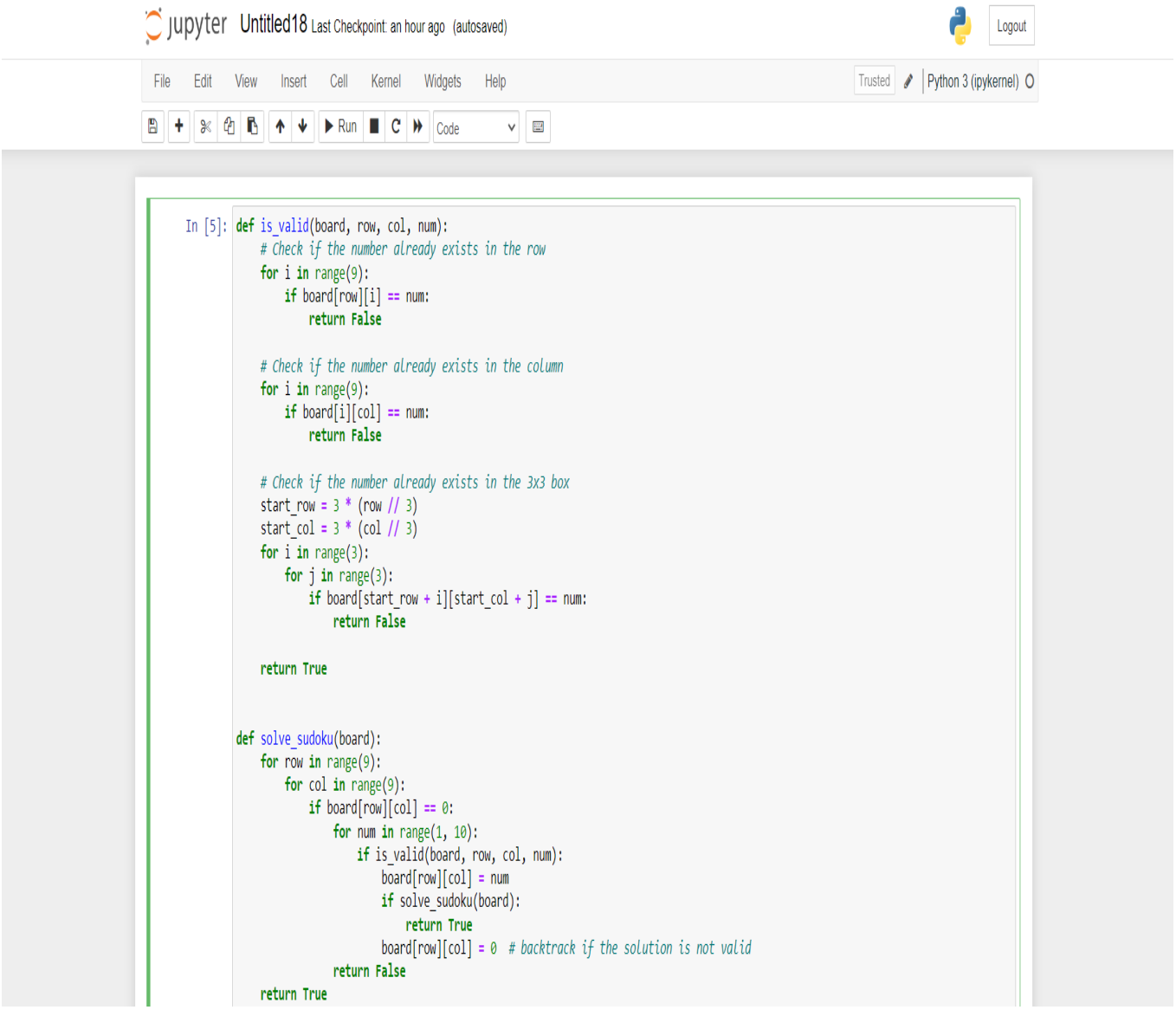
```
def solve_sudoku(board):
    for row in range(9):
        for col in range(9):
            if board[row][col] == 0:
                for num in range(1, 10):
                    if is_valid(board, row, col, num):
                        board[row][col] = num
                        if solve_sudoku(board):
                            return True
                        board[row][col] = 0  # backtrack if the solution is not valid
                return False
    return True


def print_board(board):
    for row in board:
        print(' '.join(str(num) for num in row))

# Example puzzle (0 represents empty cells)
puzzle = [
    [3, 0, 6, 5, 0, 8, 4, 0, 0],
    [5, 2, 0, 0, 0, 0, 0, 0, 0],
    [0, 8, 7, 0, 0, 0, 0, 3, 1],
    [0, 0, 3, 0, 1, 0, 0, 8, 0],
    [9, 0, 0, 8, 6, 3, 0, 0, 5],
    [0, 5, 0, 0, 9, 0, 6, 0, 0],
    [1, 3, 0, 0, 0, 0, 2, 5, 0],
```

```
    [0, 0, 0, 0, 0, 0, 0, 7, 4],
    [0, 0, 5, 2, 0, 6, 3, 0, 0]
]
if solve_sudoku(puzzle):
    print("Sudoku solved:")
    print_board(puzzle)
else:
    print("No solution exists.")
```

OUTPUT:



jupyter Untitled18 Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```python
    return True


def print_board(board):
    for row in board:
        print(' '.join(str(num) for num in row))

# Example puzzle (0 represents empty cells)
puzzle = [
    [3, 0, 6, 5, 0, 8, 4, 0, 0],
    [5, 2, 0, 0, 0, 0, 0, 0, 0],
    [0, 8, 7, 0, 0, 0, 0, 3, 1],
    [0, 0, 3, 0, 1, 0, 0, 8, 0],
    [9, 0, 0, 8, 6, 3, 0, 0, 5],
    [0, 5, 0, 0, 9, 0, 6, 0, 0],
    [1, 3, 0, 0, 0, 0, 2, 5, 0],
    [0, 0, 0, 0, 0, 0, 0, 7, 4],
    [0, 0, 5, 2, 0, 6, 3, 0, 0]
]

if solve_sudoku(puzzle):
    print("Sudoku solved:")
    print_board(puzzle)
else:
    print("No solution exists.")
```

```
Sudoku solved:
3 1 6 5 7 8 4 9 2
5 2 9 1 3 4 7 6 8
4 8 7 6 2 9 5 3 1
2 6 3 4 1 5 9 8 7
9 7 4 8 6 3 1 2 5
8 5 1 7 9 2 6 4 3
1 3 8 9 4 7 2 5 6
6 9 2 3 5 1 8 7 4
7 4 5 2 8 6 3 1 9
```

Conclusion:

The Sudoku solver program provided above demonstrates an implementation of a backtracking algorithm to solve Sudoku puzzles. The program effectively checks the validity of numbers and utilizes recursion to explore different possibilities until a valid solution is found.