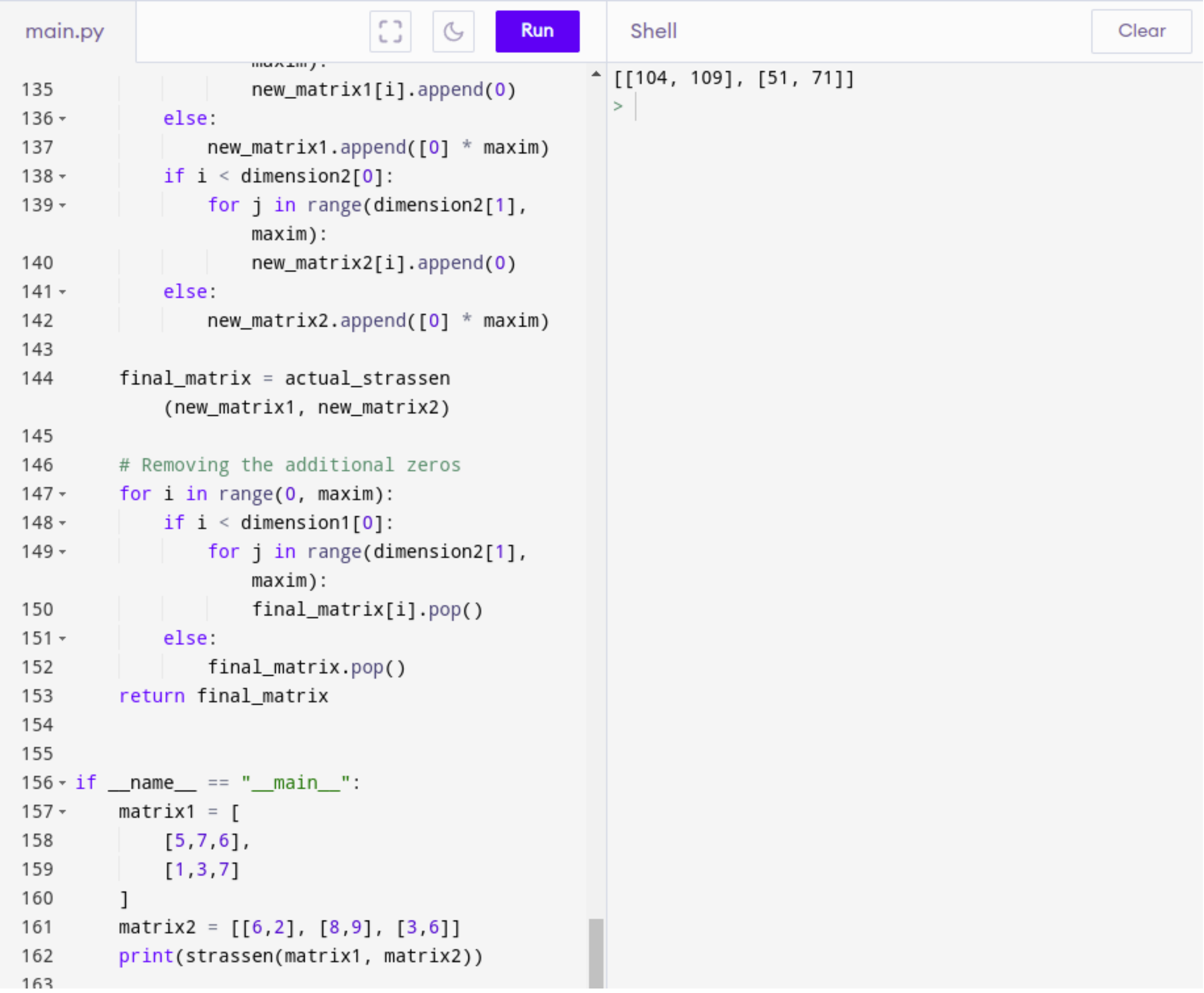
STRESSEN:

```
import math
from typing import List, Tuple


def default_matrix_multiplication(a: List, b: List) -> List:
    """
    Multiplication only for 2x2 matrices
    """
    if len(a) != 2 or len(a[0]) != 2 or len(b) != 2 or len(b[0]) != 2:
        raise Exception("Matrices are not 2x2")
    new_matrix = [
        [a[0][0] * b[0][0] + a[0][1] * b[1][0], a[0][0] * b[0][1] + a[0][1] * b[1][1]],
        [a[1][0] * b[0][0] + a[1][1] * b[1][0], a[1][0] * b[0][1] + a[1][1] * b[1][1]],
    ]
    return new_matrix


def matrix_addition(matrix_a: List, matrix_b: List):
    return [
        [matrix_a[row][col] + matrix_b[row][col] for col in range(len(matrix_a[row]))]
        for row in range(len(matrix_a))
    ]


def matrix_subtraction(matrix_a: List, matrix_b: List):
    return [
        [matrix_a[row][col] - matrix_b[row][col] for col in range(len(matrix_a[row]))]
        for row in range(len(matrix_a))
    ]


def split_matrix(a: List,) -> Tuple[List, List, List, List]:
    """
    Given an even length matrix, returns the top_left, top_right, bot_left, bot_right
    quadrant.

    >>> split_matrix([[4,3,2,4],[2,3,1,1],[6,5,4,3],[8,4,1,6]])
    ([[4, 3], [2, 3]], [[2, 4], [1, 1]], [[6, 5], [8, 4]], [[4, 3], [1, 6]])
    >>> split_matrix([
    ...     [4,3,2,4,4,3,2,4],[2,3,1,1,2,3,1,1],[6,5,4,3,6,5,4,3],[8,4,1,6,8,4,1,6],
    ...     [4,3,2,4,4,3,2,4],[2,3,1,1,2,3,1,1],[6,5,4,3,6,5,4,3],[8,4,1,6,8,4,1,6]
    ... ])  # doctest: +NORMALIZE_WHITESPACE
    ([[4, 3, 2, 4], [2, 3, 1, 1], [6, 5, 4, 3], [8, 4, 1, 6]], [[4, 3, 2, 4],
      [2, 3, 1, 1], [6, 5, 4, 3], [8, 4, 1, 6]], [[4, 3, 2, 4], [2, 3, 1, 1],
      [6, 5, 4, 3], [8, 4, 1, 6]], [[4, 3, 2, 4], [2, 3, 1, 1], [6, 5, 4, 3],
      [8, 4, 1, 6]])
    """
    if len(a) % 2 != 0 or len(a[0]) % 2 != 0:
        raise Exception("Odd matrices are not supported!")

    matrix_length = len(a)
    mid = matrix_length // 2

    top_right = [[a[i][j] for j in range(mid, matrix_length)] for i in range(mid)]
    bot_right = [
        [a[i][j] for j in range(mid, matrix_length)] for i in range(mid, matrix_length)
    ]

    top_left = [[a[i][j] for j in range(mid)] for i in range(mid)]
    bot_left = [[a[i][j] for j in range(mid)] for i in range(mid, matrix_length)]

    return top_left, top_right, bot_left, bot_right


def matrix_dimensions(matrix: List) -> Tuple[int, int]:
    return len(matrix), len(matrix[0])



def print_matrix(matrix: List) -> None:
    for i in range(len(matrix)):
        print(matrix[i])



def actual_strassen(matrix_a: List, matrix_b: List) -> List:
    """
    Recursive function to calculate the product of two matrices, using the Strassen
    Algorithm.  It only supports even length matrices.
    """
    if matrix_dimensions(matrix_a) == (2, 2):
        return default_matrix_multiplication(matrix_a, matrix_b)

    a, b, c, d = split_matrix(matrix_a)
    e, f, g, h = split_matrix(matrix_b)

    t1 = actual_strassen(a, matrix_subtraction(f, h))
    t2 = actual_strassen(matrix_addition(a, b), h)
    t3 = actual_strassen(matrix_addition(c, d), e)
    t4 = actual_strassen(d, matrix_subtraction(g, e))
    t5 = actual_strassen(matrix_addition(a, d), matrix_addition(e, h))
    t6 = actual_strassen(matrix_subtraction(b, d), matrix_addition(g, h))
    t7 = actual_strassen(matrix_subtraction(a, c), matrix_addition(e, f))

    top_left = matrix_addition(matrix_subtraction(matrix_addition(t5, t4), t2), t6)
    top_right = matrix_addition(t1, t2)
    bot_left = matrix_addition(t3, t4)
    bot_right = matrix_subtraction(matrix_subtraction(matrix_addition(t1, t5), t3), t7)

    # construct the new matrix from our 4 quadrants
    new_matrix = []
    for i in range(len(top_right)):
        new_matrix.append(top_left[i] + top_right[i])
    for i in range(len(bot_right)):
        new_matrix.append(bot_left[i] + bot_right[i])
    return new_matrix


def strassen(matrix1: List, matrix2: List) -> List:
    """
    >>> strassen([[2,1,3],[3,4,6],[1,4,2],[7,6,7]], [[4,2,3,4],[2,1,1,1],[8,6,4,2]])
    [[34, 23, 19, 15], [68, 46, 37, 28], [28, 18, 15, 12], [96, 62, 55, 48]]
    >>> strassen([[3,7,5,6,9],[1,5,3,7,8],[1,4,4,5,7]], [[2,4],[5,2],[1,7],[5,5],[7,8]])
    [[139, 163], [121, 134], [100, 121]]
    """
    if matrix_dimensions(matrix1)[1] != matrix_dimensions(matrix2)[0]:
        raise Exception(
            f"Unable to multiply these matrices, please check the dimensions. \n"
            f"Matrix A:{matrix1} \nMatrix B:{matrix2}"
        )
    dimension1 = matrix_dimensions(matrix1)
    dimension2 = matrix_dimensions(matrix2)

    if dimension1[0] == dimension1[1] and dimension2[0] == dimension2[1]:
        return matrix1, matrix2

    maximum = max(max(dimension1), max(dimension2))
    maxim = int(math.pow(2, math.ceil(math.log2(maximum))))
    new_matrix1 = matrix1
    new_matrix2 = matrix2

    # Adding zeros to the matrices so that the arrays dimensions are the same and also
    # power of 2
    for i in range(0, maxim):
        if i < dimension1[0]:
            for j in range(dimension1[1], maxim):
                new_matrix1[i].append(0)
        else:
            new_matrix1.append([0] * maxim)
        if i < dimension2[0]:
            for j in range(dimension2[1], maxim):
                new_matrix2[i].append(0)
        else:
            new_matrix2.append([0] * maxim)
    final_matrix = actual_strassen(new_matrix1, new_matrix2)

    # Removing the additional zeros
    for i in range(0, maxim):
        if i < dimension1[0]:
            for j in range(dimension2[1], maxim):
                final_matrix[i].pop()
        else:
            final_matrix.pop()
    return final_matrix


if __name__ == "__main__":
    matrix1 = [
        [5,7,6],
        [1,3,7]
    ]
    matrix2 = [[6,2], [8,9], [3,6]]
    print(strassen(matrix1, matrix2))
```

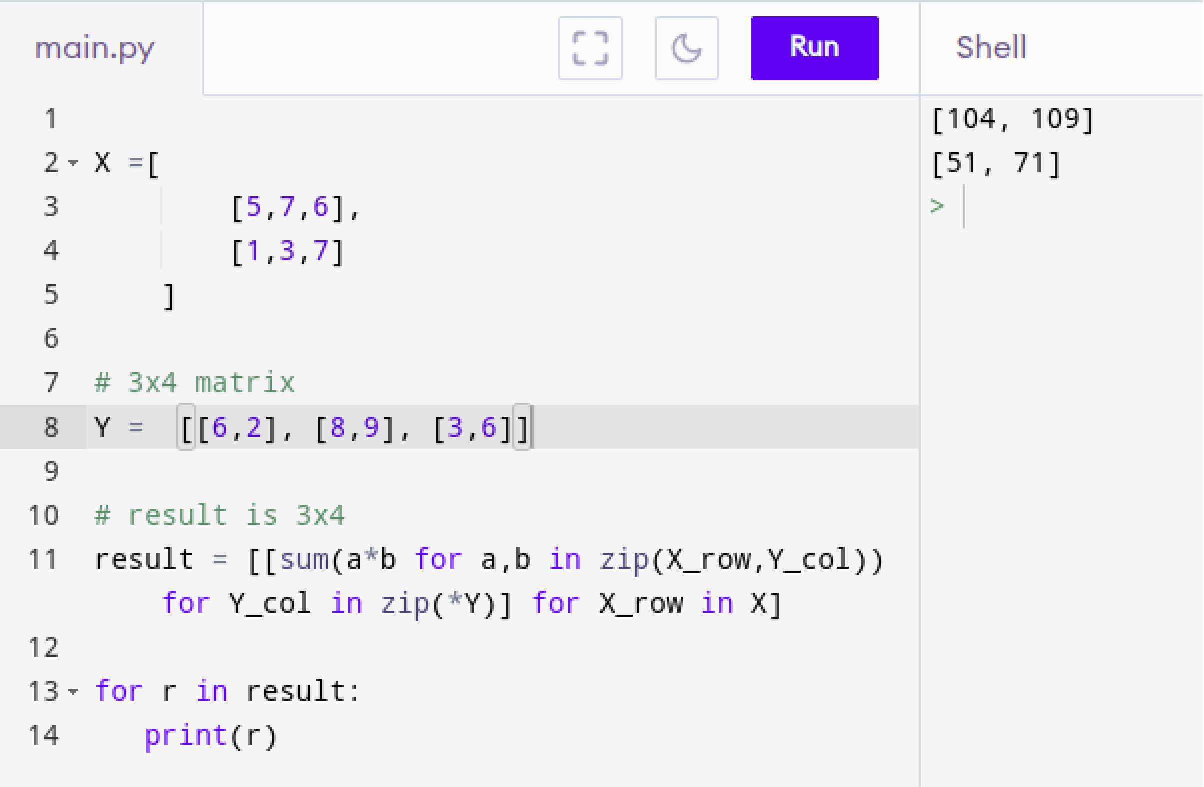


```
                maxim):
                new_matrix1[i].append(0)
        else:
            new_matrix1.append([0] * maxim)
        if i < dimension2[0]:
            for j in range(dimension2[1], maxim):
                new_matrix2[i].append(0)
        else:
            new_matrix2.append([0] * maxim)

    final_matrix = actual_strassen(new_matrix1, new_matrix2)

    # Removing the additional zeros
    for i in range(0, maxim):
        if i < dimension1[0]:
            for j in range(dimension2[1], maxim):
                final_matrix[i].pop()
        else:
            final_matrix.pop()
    return final_matrix


if __name__ == "__main__":
    matrix1 = [
        [5,7,6],
        [1,3,7]
    ]
    matrix2 = [[6,2], [8,9], [3,6]]
    print(strassen(matrix1, matrix2))
```

```
[[104, 109], [51, 71]]
>
```

normal:

```
X =[
        [5,7,6],
        [1,3,7]
    ]

# 3x4 matrix
Y =  [[6,2], [8,9], [3,6]]

# result is 3x4
result = [[sum(a*b for a,b in zip(X_row,Y_col)) for Y_col in zip(*Y)] for X_row in X]

for r in result:
   print(r)
```



```python

X =[
        [5,7,6],
        [1,3,7]
    ]

# 3x4 matrix
Y =  [[6,2], [8,9], [3,6]]

# result is 3x4
result = [[sum(a*b for a,b in zip(X_row,Y_col)) for Y_col in zip(*Y)] for X_row in X]

for r in result:
   print(r)
```

```
[104, 109]
[51, 71]
> 
```