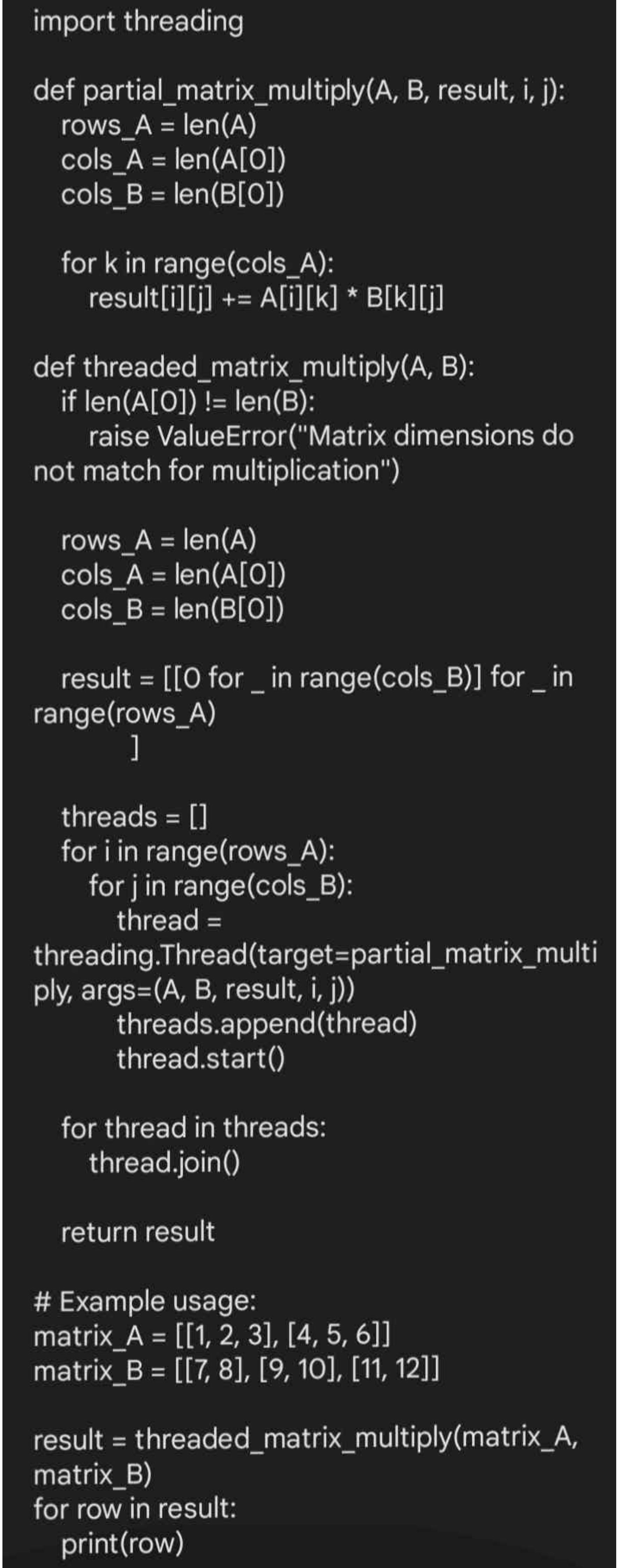
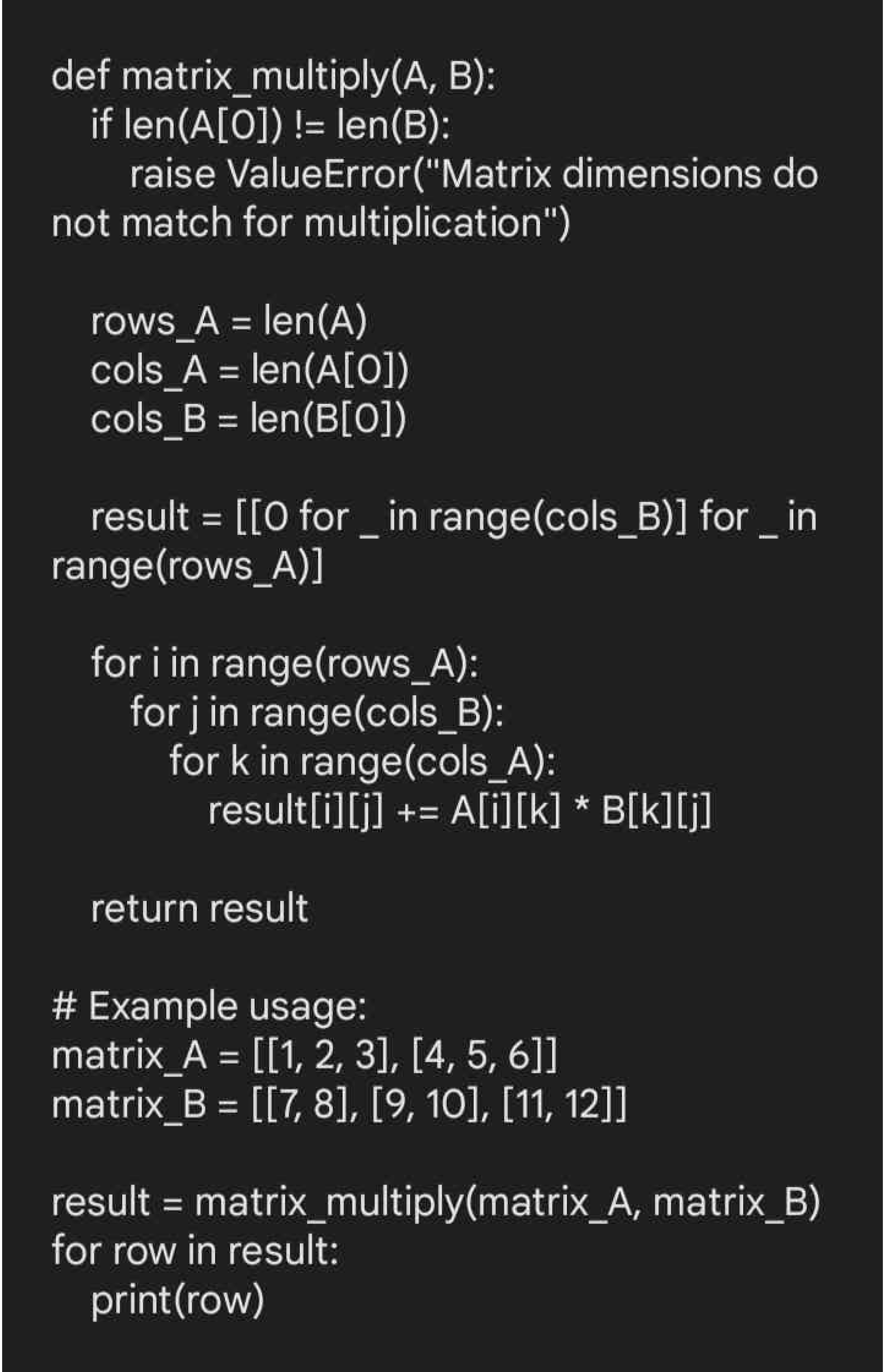


```
import threading

def partial_matrix_multiply(A, B, result, i, j):
    rows_A = len(A)
    cols_A = len(A[0])
    cols_B = len(B[0])

    for k in range(cols_A):
        result[i][j] += A[i][k] * B[k][j]

def threaded_matrix_multiply(A, B):
    if len(A[0]) != len(B):
        raise ValueError("Matrix dimensions do
not match for multiplication")

    rows_A = len(A)
    cols_A = len(A[0])
    cols_B = len(B[0])

    result = [[0 for _ in range(cols_B)] for _ in
range(rows_A)
              ]

    threads = []
    for i in range(rows_A):
        for j in range(cols_B):
            thread =
threading.Thread(target=partial_matrix_multi
ply, args=(A, B, result, i, j))
            threads.append(thread)
            thread.start()

    for thread in threads:
        thread.join()

    return result

# Example usage:
matrix_A = [[1, 2, 3], [4, 5, 6]]
matrix_B = [[7, 8], [9, 10], [11, 12]]

result = threaded_matrix_multiply(matrix_A,
matrix_B)
for row in result:
    print(row)
```



```
def matrix_multiply(A, B):
    if len(A[0]) != len(B):
        raise ValueError("Matrix dimensions do
not match for multiplication")

    rows_A = len(A)
    cols_A = len(A[0])
    cols_B = len(B[0])

    result = [[0 for _ in range(cols_B)] for _ in
range(rows_A)]

    for i in range(rows_A):
        for j in range(cols_B):
            for k in range(cols_A):
                result[i][j] += A[i][k] * B[k][j]

    return result

# Example usage:
matrix_A = [[1, 2, 3], [4, 5, 6]]
matrix_B = [[7, 8], [9, 10], [11, 12]]

result = matrix_multiply(matrix_A, matrix_B)
for row in result:
    print(row)
```