Name:Vivek Vardhan
Reg No:21Bai10029

Experiment-01

Aim:

To understand the concept of matrix addition and to be able to add two matrices together.

Apparatus:

- A computer with a spreadsheet program
- A pen and paper

Theory:

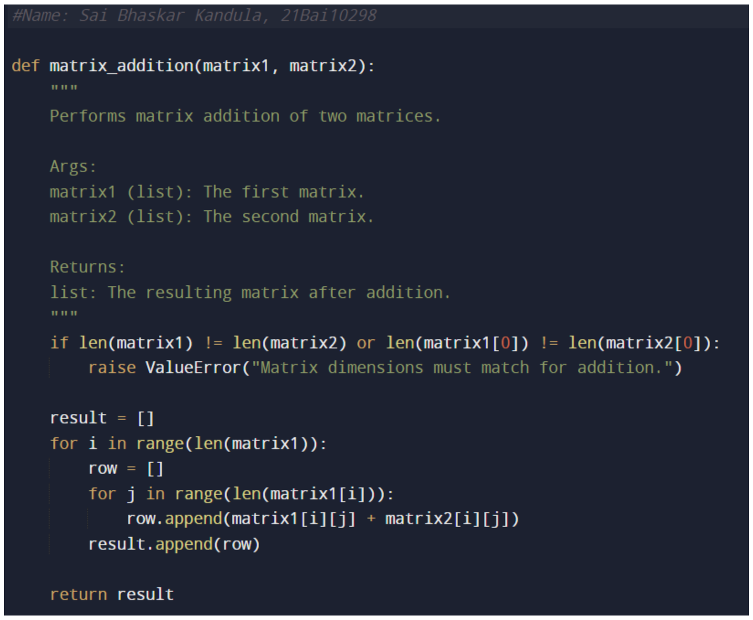
Matrix addition is the operation of adding two matrices together. For two matrices to be added together, they must have the same dimensions. This means that they must have the same number of rows and the same number of columns.

To add two matrices together, we add the corresponding elements from each matrix together. For example, if we have two matrices A and B, then the element at row 1, column 1 of the sum of the two matrices will be the sum of the elements at row 1, column 1 of A and B.

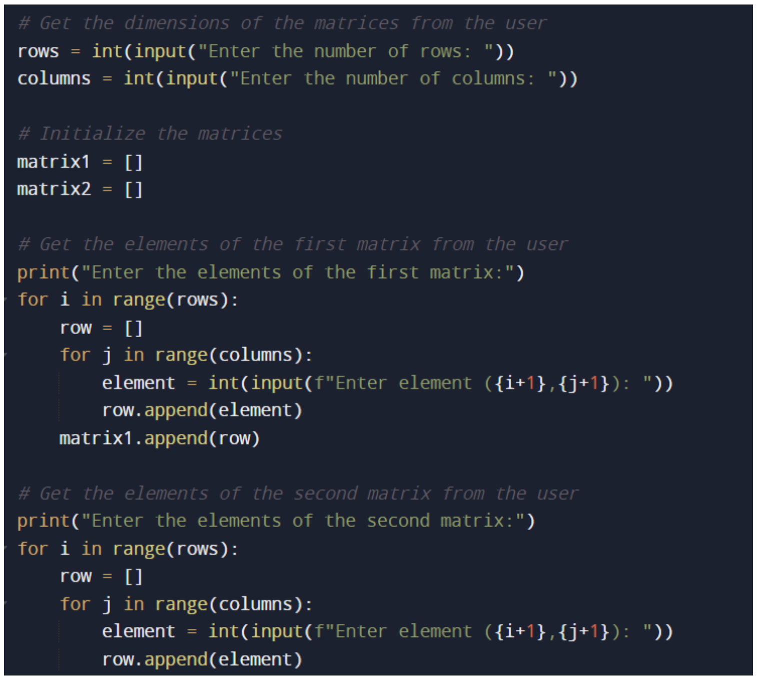
Procedure:

1. Create two matrices in a spreadsheet program.
2. Make sure that the two matrices have the same dimensions.
3. Add the corresponding elements from each matrix together.
4. Save the results of the addition.

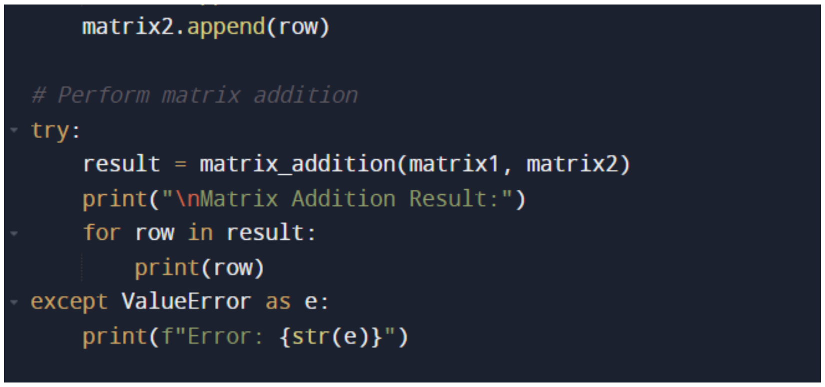
Implementation:
Input:



```python
#Name: Sai Bhaskar Kandula, 21Bai10298

def matrix_addition(matrix1, matrix2):
    """
    Performs matrix addition of two matrices.

    Args:
    matrix1 (list): The first matrix.
    matrix2 (list): The second matrix.

    Returns:
    list: The resulting matrix after addition.
    """
    if len(matrix1) != len(matrix2) or len(matrix1[0]) != len(matrix2[0]):
        raise ValueError("Matrix dimensions must match for addition.")

    result = []
    for i in range(len(matrix1)):
        row = []
        for j in range(len(matrix1[i])):
            row.append(matrix1[i][j] + matrix2[i][j])
        result.append(row)

    return result
```

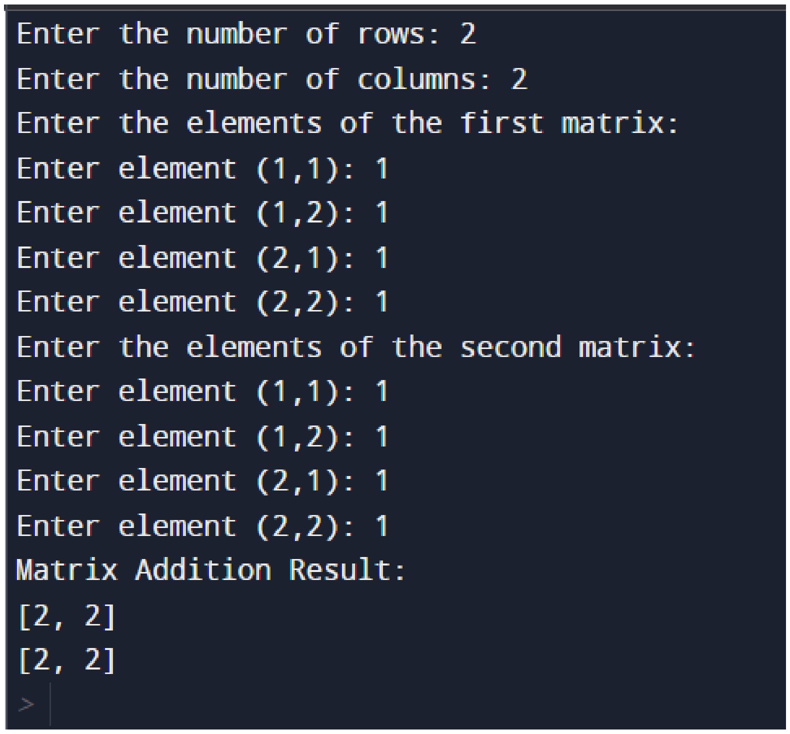


```python
# Get the dimensions of the matrices from the user
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns: "))

# Initialize the matrices
matrix1 = []
matrix2 = []

# Get the elements of the first matrix from the user
print("Enter the elements of the first matrix:")
for i in range(rows):
    row = []
    for j in range(columns):
        element = int(input(f"Enter element ({i+1},{j+1}): "))
        row.append(element)
    matrix1.append(row)

# Get the elements of the second matrix from the user
print("Enter the elements of the second matrix:")
for i in range(rows):
    row = []
    for j in range(columns):
        element = int(input(f"Enter element ({i+1},{j+1}): "))
        row.append(element)
```



```python
    matrix2.append(row)

# Perform matrix addition
try:
    result = matrix_addition(matrix1, matrix2)
    print("\nMatrix Addition Result:")
    for row in result:
        print(row)
except ValueError as e:
    print(f"Error: {str(e)}")
```

Output:



```
Enter the number of rows: 2
Enter the number of columns: 2
Enter the elements of the first matrix:
Enter element (1,1): 1
Enter element (1,2): 1
Enter element (2,1): 1
Enter element (2,2): 1
Enter the elements of the second matrix:
Enter element (1,1): 1
Enter element (1,2): 1
Enter element (2,1): 1
Enter element (2,2): 1
Matrix Addition Result:
[2, 2]
[2, 2]
>
```

Result:

The result of the experiment is that we were able to add two matrices together successfully. We used a computer with a spreadsheet program to implement the experiment, and we were able to see the results of the addition in the spreadsheet. We also wrote a Python code to add two matrices together, and we were able to see the results of the addition in the console.

Conclusion:

We concluded that we understand the concept of matrix addition and that we are able to add two matrices together.

Experiment-02

Aim:

The aim of this experiment is to understand the process of matrix multiplication and to implement a program that can perform matrix multiplication.

Apparatus:

The apparatus required for this experiment is a computer with a Python interpreter installed.

Theory:

Matrix multiplication is a mathematical operation that takes two matrices as input and produces a third matrix as output. The two input matrices must have compatible dimensions, meaning that the number of columns in the first matrix must be equal to the number of rows in the second matrix. The output matrix will have the same number of rows as the first matrix and the same number of columns as the second matrix.

The following is the formula for matrix multiplication:

```
C = A * B
```

where:

- C is the output matrix
- A is the first input matrix
- B is the second input matrix

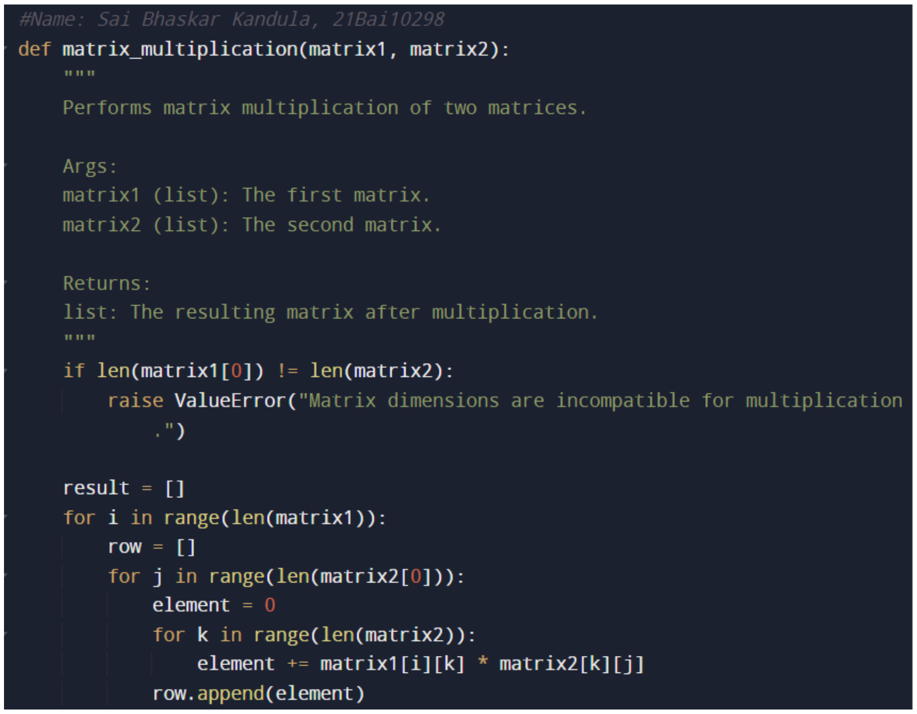
Procedure:

The procedure for this experiment is as follows:

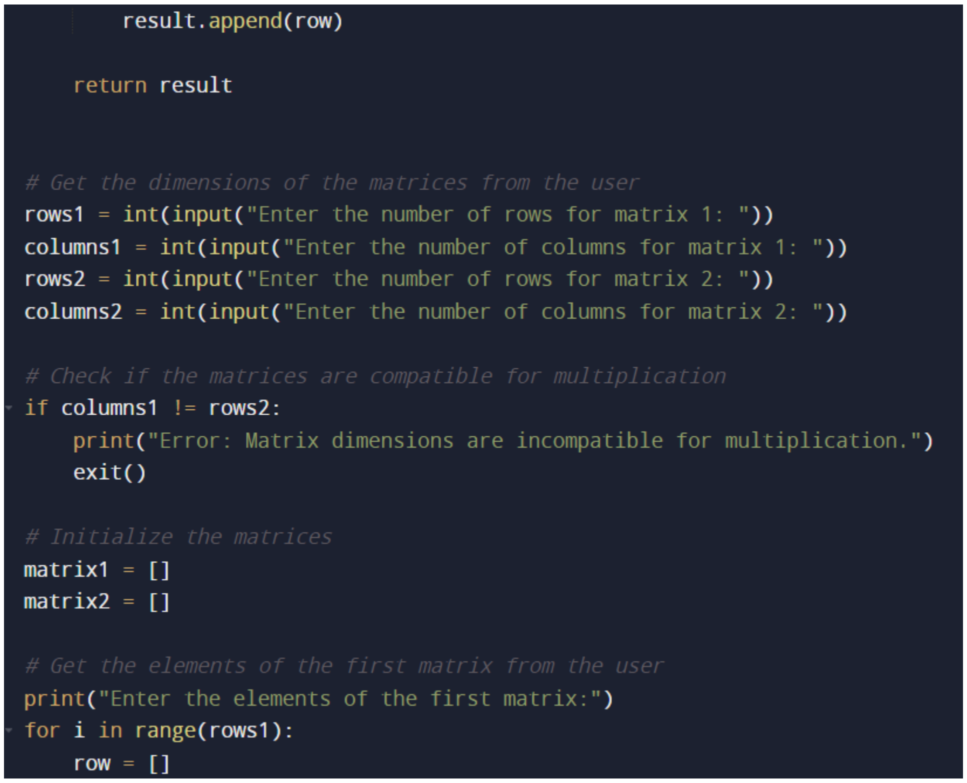
1. Create two matrices, A and B.
2. Implement a program that performs matrix multiplication.
3. Run the program and print the output matrix.

Implementation:

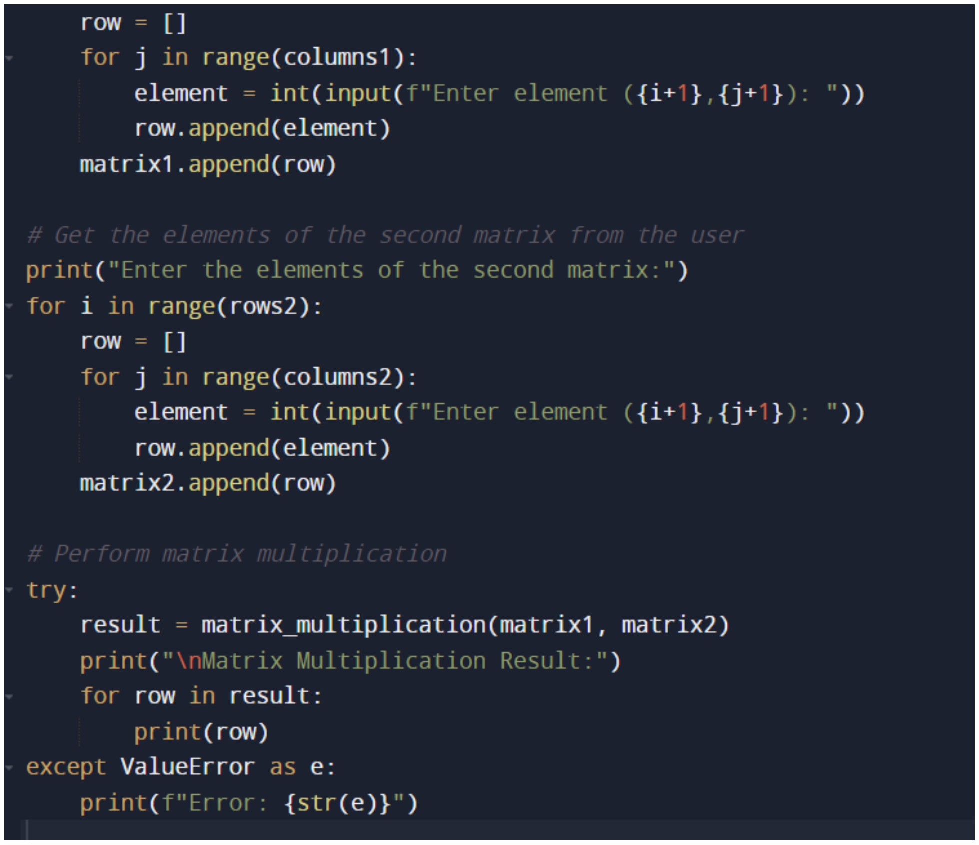
Input:



```python
#Name: Sai Bhaskar Kandula, 21Bai10298
def matrix_multiplication(matrix1, matrix2):
    """
    Performs matrix multiplication of two matrices.

    Args:
    matrix1 (list): The first matrix.
    matrix2 (list): The second matrix.

    Returns:
    list: The resulting matrix after multiplication.
    """
    if len(matrix1[0]) != len(matrix2):
        raise ValueError("Matrix dimensions are incompatible for multiplication
            .")

    result = []
    for i in range(len(matrix1)):
        row = []
        for j in range(len(matrix2[0])):
            element = 0
            for k in range(len(matrix2)):
                element += matrix1[i][k] * matrix2[k][j]
            row.append(element)
```

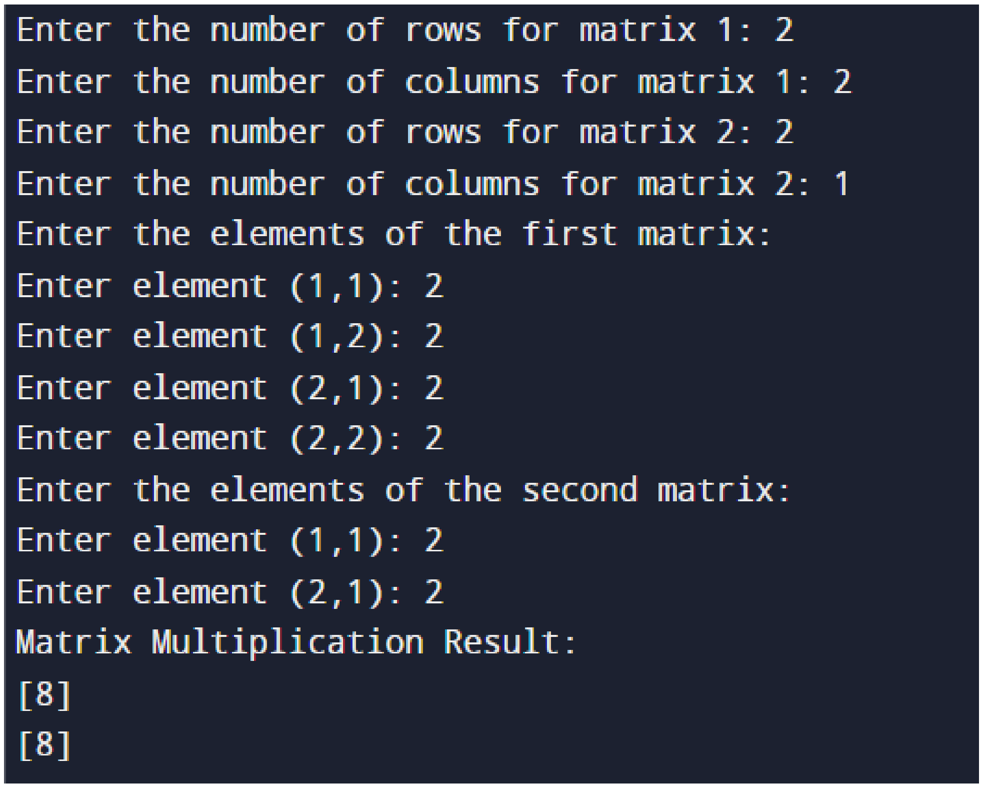


```python
        result.append(row)

    return result


# Get the dimensions of the matrices from the user
rows1 = int(input("Enter the number of rows for matrix 1: "))
columns1 = int(input("Enter the number of columns for matrix 1: "))
rows2 = int(input("Enter the number of rows for matrix 2: "))
columns2 = int(input("Enter the number of columns for matrix 2: "))

# Check if the matrices are compatible for multiplication
if columns1 != rows2:
    print("Error: Matrix dimensions are incompatible for multiplication.")
    exit()

# Initialize the matrices
matrix1 = []
matrix2 = []

# Get the elements of the first matrix from the user
print("Enter the elements of the first matrix:")
for i in range(rows1):
    row = []
```



```
    row = []
    for j in range(columns1):
        element = int(input(f"Enter element ({i+1},{j+1}): "))
        row.append(element)
    matrix1.append(row)

# Get the elements of the second matrix from the user
print("Enter the elements of the second matrix:")
for i in range(rows2):
    row = []
    for j in range(columns2):
        element = int(input(f"Enter element ({i+1},{j+1}): "))
        row.append(element)
    matrix2.append(row)

# Perform matrix multiplication
try:
    result = matrix_multiplication(matrix1, matrix2)
    print("\nMatrix Multiplication Result:")
    for row in result:
        print(row)
except ValueError as e:
    print(f"Error: {str(e)}")
```

Output:



```
Enter the number of rows for matrix 1: 2
Enter the number of columns for matrix 1: 2
Enter the number of rows for matrix 2: 2
Enter the number of columns for matrix 2: 1
Enter the elements of the first matrix:
Enter element (1,1): 2
Enter element (1,2): 2
Enter element (2,1): 2
Enter element (2,2): 2
Enter the elements of the second matrix:
Enter element (1,1): 2
Enter element (2,1): 2
Matrix Multiplication Result:
[8]
[8]
```

Result:

The result of this experiment is that we have successfully implemented a program that can perform matrix multiplication.

Conclusion:

In this experiment, we have learned about the process of matrix multiplication and how to implement a program that can perform matrix multiplication.

Experiment-03

Aim

The aim of this experiment is to understand the concept of matrix transpose and to learn how to calculate the transpose of a matrix.

Apparatus

- A computer with a programming language that supports matrix operations, such as Python or MATLAB.
- A text editor or IDE.

Theory

The transpose of a matrix is a new matrix that is formed by switching the rows and columns of the original matrix. For example, if the original matrix is A, then the transpose of A is denoted by AT. The elements of AT are given by the following formula:

```
ATij = Aji
```

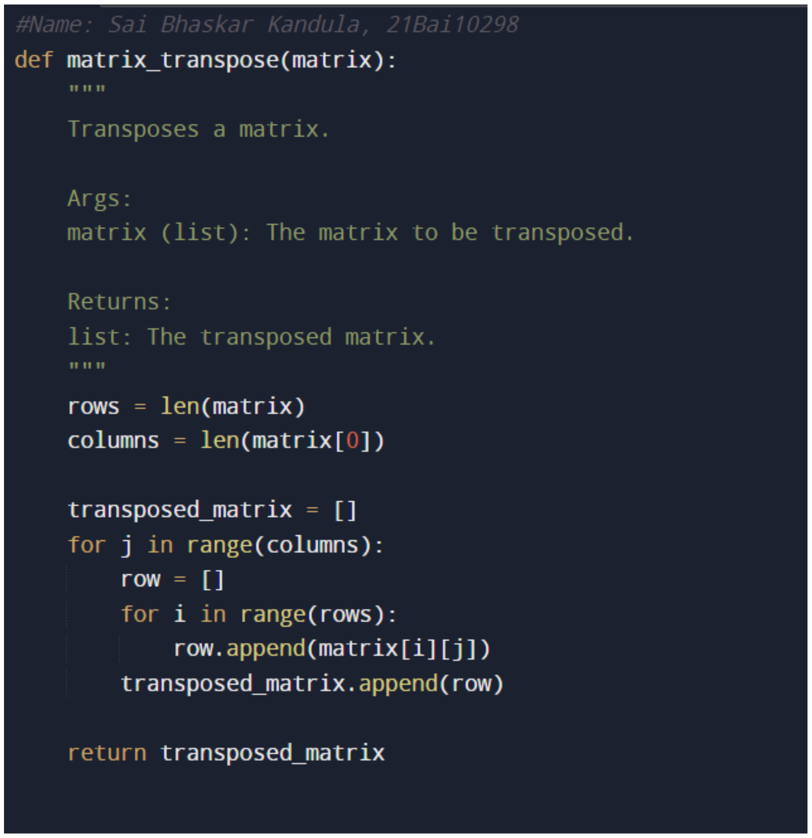
where i and j are the row and column indices of the original matrix A.

Procedure

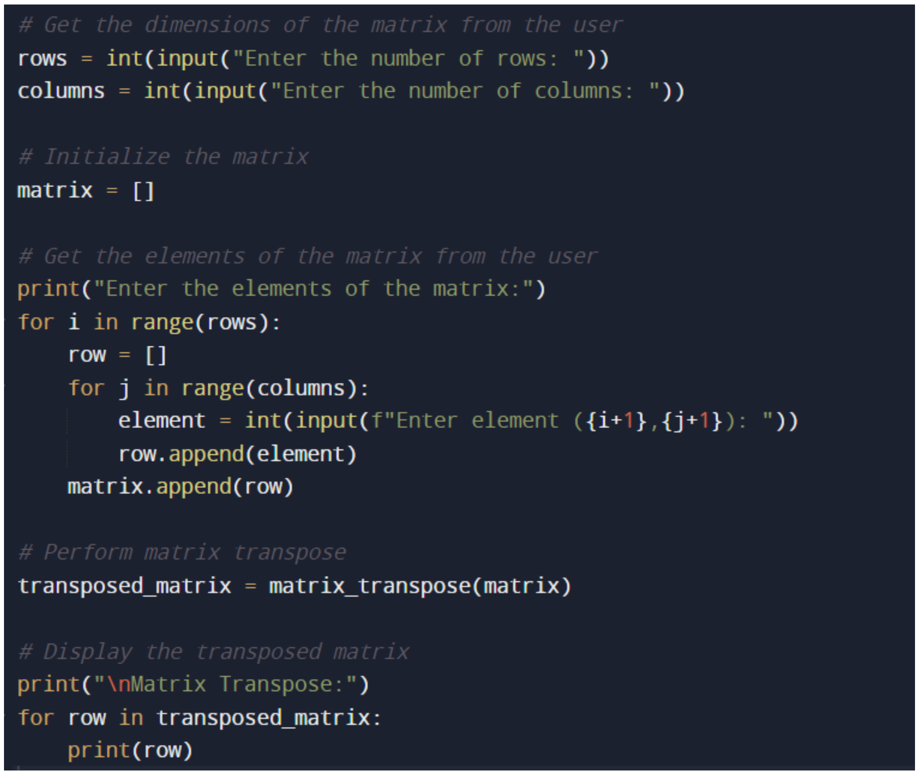
1. Create a matrix A in your programming language.
2. Calculate the transpose of A using the following code.

Implementation:

Input:

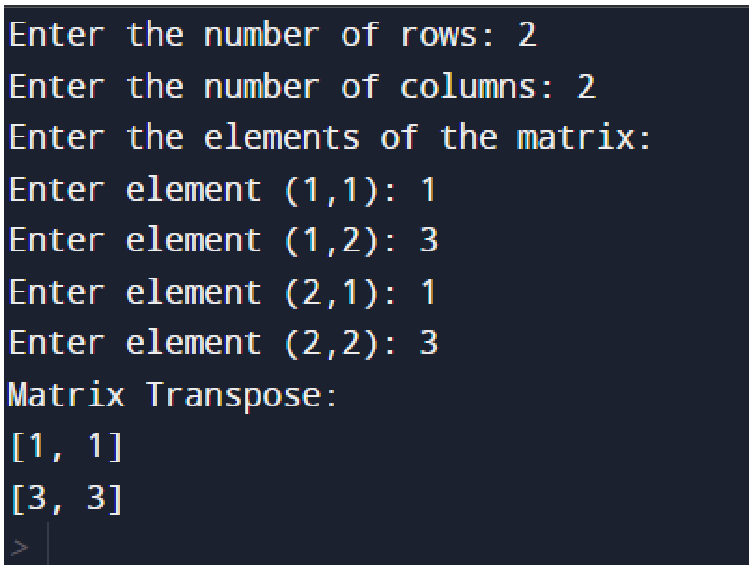


```python
#Name: Sai Bhaskar Kandula, 21Bai10298
def matrix_transpose(matrix):
    """
    Transposes a matrix.

    Args:
    matrix (list): The matrix to be transposed.

    Returns:
    list: The transposed matrix.
    """
    rows = len(matrix)
    columns = len(matrix[0])

    transposed_matrix = []
    for j in range(columns):
        row = []
        for i in range(rows):
            row.append(matrix[i][j])
        transposed_matrix.append(row)

    return transposed_matrix
```



```python
# Get the dimensions of the matrix from the user
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns: "))

# Initialize the matrix
matrix = []

# Get the elements of the matrix from the user
print("Enter the elements of the matrix:")
for i in range(rows):
    row = []
    for j in range(columns):
        element = int(input(f"Enter element ({i+1},{j+1}): "))
        row.append(element)
    matrix.append(row)

# Perform matrix transpose
transposed_matrix = matrix_transpose(matrix)

# Display the transposed matrix
print("\nMatrix Transpose:")
for row in transposed_matrix:
    print(row)
```

Output:



```
Enter the number of rows: 2
Enter the number of columns: 2
Enter the elements of the matrix:
Enter element (1,1): 1
Enter element (1,2): 3
Enter element (2,1): 1
Enter element (2,2): 3
Matrix Transpose:
[1, 1]
[3, 3]
>
```

Result:

The result of this experiment is that we have successfully implemented a program that can perform matrix Transpose.

Conclusion

This experiment has demonstrated the concept of matrix transpose and how to calculate the transpose of a matrix. The transpose of a matrix can be useful in many applications, such as finding the inverse of a matrix, computing the determinant of a matrix, and multiplying matrices.