```
# -*- coding: utf-8 -*-
"""PPA5.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1OLqHvqXvdVRsB1S9vj8LWjeGwJAE4dre
"""

def print_matrix(matrix):
    for row in matrix:
        print(row)
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
print("Matrix:")
print_matrix(matrix)

def add_matrices(matrix1, matrix2):
    result = []
    for i in range(len(matrix1)):
        row = []
        for j in range(len(matrix1[0])):
            row.append(matrix1[i][j] + matrix2[i][j])
        result.append(row)
    return result

def print_matrix(matrix):
    for row in matrix:
        print(row)
matrix1 = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

matrix2 = [
    [9, 8, 7],
    [6, 5, 4],
    [3, 2, 1]
]
result_matrix = add_matrices(matrix1, matrix2)
print("Matrix 1:")
print_matrix(matrix1)
print("\nMatrix 2:")
print_matrix(matrix2)
print("\nResult of Addition:")
print_matrix(result_matrix)

def multiply_matrices(matrix1, matrix2):
    result = []
    rows1 = len(matrix1)
    cols1 = len(matrix1[0])
    rows2 = len(matrix2)
    cols2 = len(matrix2[0])


    if cols1 != rows2:
        print("Multiplication not possible. Number of columns in matrix1 must be equal to number of rows in matrix2.")
        return None


    for i in range(rows1):
        row = []
        for j in range(cols2):
            row.append(0)
        result.append(row)


    for i in range(rows1):
        for j in range(cols2):
            for k in range(cols1):
                result[i][j] += matrix1[i][k] * matrix2[k][j]

    return result
```

```
def print_matrix(matrix):
    for row in matrix:
        print(row)

matrix1 = [
    [1, 2, 3],
    [4, 5, 6]
]

matrix2 = [
    [7, 8],
    [9, 10],
    [11, 12]
]
result_matrix = multiply_matrices(matrix1, matrix2)
if result_matrix:
    print("Matrix 1:")
    print_matrix(matrix1)
    print("\nMatrix 2:")
    print_matrix(matrix2)
    print("\nResult of Multiplication:")
    print_matrix(result_matrix)

import math

def area_square(side):
    """Calculate the area of a square."""
    return side * side

def perimeter_square(side):
    """Calculate the perimeter of a square."""
    return 4 * side

def area_rectangle(length, width):
    """Calculate the area of a rectangle."""
    return length * width

def perimeter_rectangle(length, width):
    """Calculate the perimeter of a rectangle."""
    return 2 * (length + width)

def area_circle(radius):
    """Calculate the area of a circle."""
    return math.pi * radius**2

def circumference_circle(radius):
    """Calculate the circumference of a circle."""
    return 2 * math.pi * radius

import geometry

side_length = 5
print("Area of square:", geometry.area_square(side_length))
print("Perimeter of square:", geometry.perimeter_square(side_length))
length = 4
width = 6
print("Area of rectangle:", geometry.area_rectangle(length, width))
print("Perimeter of rectangle:", geometry.perimeter_rectangle(length, width)
radius = 3
print("Area of circle:", geometry.area_circle(radius))
print("Circumference of circle:", geometry.circumference_circle(radius))



import geometry

try:
    side_length = 5
    print("Area of square:", geometry.area_square(side_length))
    print("Perimeter of square:", geometry.perimeter_square(side_length))

    length = 4
    width = 6
    print("Area of rectangle:", geometry.area_rectangle(length, width))
    print("Perimeter of rectangle:", geometry.perimeter_rectangle(length, width))
    radius = 3
    print("Area of circle:", geometry.area_circle(radius))
    print("Circumference of circle:", geometry.circumference_circle(radius))

    print("Testing invalid input:")
    print(geometry.area_square(-1))  # Should raise ValueError
    print(geometry.area_rectangle(3, -4))  # Should raise ValueError
    print(geometry.area_circle(-2))  # Should raise ValueError

except ValueError as e:
    print("Error:", e)
```