**Assignment C**

## Implement a set of classes to provide:

## 2d vector/point coordinate

## 3x3 matrix and functions associated to it:

## Inverse

## Multiply

## Transform point/vector

## Functions to create translation/rotation/scaling matrix

## **2d vector/point coordinate**

**Prerequisites:**

**ANACONDA Spyder**

1. Import math

**Source code:**



**Note:** Kindly open the above .txt file to review the source code

**Output:**

## Example usage

p1 = Point2D(0, 0)

p2 = Point2D(3, 4)

v = p2 - p1 # Vector2D object

print(v.magnitude()) # 5.0

v.normalize()

print(v.magnitude()) # 1.0

p3 = p1 + v \* 5 # Point2D object

print(p3.x, p3.y) # (3.0, 4.0)

print(p1.distance\_to(p2)) # 5.0

* Two “point2D” objects, ‘P1’ and ‘P2’ represent points at coordinates (0,0) and (3,4) respectively
* “vector2D” object represents the vector from ‘P1’ and ‘P2’ and calculated the magnitude and normalized.

## **3x3 matrix and functions associated to it:**

## **• Inverse**

## **• multiply**

## **• transform point/vector**

**Prerequisites:**

**ANACONDA Spyder**

**Source code:**



**Note:** Kindly open the above .txt file to review the source code

**Output:**

a = Matrix3x3([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

b = Matrix3x3([[9, 8, 7], [6, 5, 4], [3, 2, 1]])

c = a \* b # multiply two matrices

print(c.elements) # [[30, 24, 18], [84, 69, 54], [138, 114, 90]]

point = (1, 2, 3)

vector = (4, 5, 6)

a.transform\_point(point) # (14, 32, 50)

a.transform\_vector(vector) # (32, 77, 122)

## **Functions to create translation/rotation/scaling matrix**

**Prerequisites:**

**ANACONDA Spyder**

1. Import math

**Source code:**

****

**Note:** Kindly open the above .txt file to review the source code

**Output:**

The code displays original and translated, rotated, and scaled matrix

**Leverage this library to animate the watch position by combining a set of matrices:  
the watch shall rotate on itself, scale up and down, and rotate around an arbitrary point (randomly defined at page load or from a user input field)**

**Prerequisites:**

**ANACONDA Spyder**

1. Install tkinter library using “pip install tk”
2. Install PIL library
3. Import math
4. Import numpy

**Implementation steps**

1. Install the required libraries
2. Import the required modules
3. Watch animation class is created and watch image is given as the input
4. Function such as get\_rotation\_matriz, apply\_transformation, and scale\_image is created
5. Animation function is also created to automatic rotate of the image which applies the desired transformations tot eh watch display using matrix operations
6. The output of this code is to rotates the uploaded image around the specified rotation center, scales up and down based on the current rotation angle, and updates the watch display on the canvas. The animation is achieved by continuously updating the rotation angle and scale.

**Source code:**

****

**Output**

|  |
| --- |
|  |
| (a) |
|  |
| (b) |
| Figure: a) Representative image b) animation of the given image by continuous rotating, scaling around the arbitrary point |