Amit Divekar | Practical 3

Computational Geometry Basics with SymPy

In [1]:
```
from sympy import Point, Line, Segment, Triangle, Polygon, pi, sqrt
import sympy as sp
```

Q1. Write a Python program using the SymPy library to find the distance between two points in a 2D plane. Define two points P1(2,3) and P2(6,7)

In [2]:
```
from sympy import Point
P1 = Point(2, 3)
P2 = Point(6, 7)
distance = P1.distance(P2)
print("Point P1:", P1)
print("Point P2:", P2)
print("Distance between P1 and P2:", distance)
```

```
Point P1: Point2D(2, 3)
Point P2: Point2D(6, 7)
Distance between P1 and P2: 4*sqrt(2)
```

Q2. Write a Python program using the SymPy library to represent a line and a line segment between two points A(1,1) and B(5,4).

In [3]:
```
from sympy import Point, Line, Segment
A = Point(1, 1)
B = Point(5, 4)
line = Line(A, B)
segment = Segment(A, B)
print("Line equation:", line)
print("Segment length:", segment.length)
print("Slope of the line:", line.slope)
```

```
Line equation: Line2D(Point2D(1, 1), Point2D(5, 4))
Segment length: 5
Slope of the line: 3/4
```

Q3. Write a Python program using the SymPy library to define a triangle in a 2D plane with three points P1(0,0), P2(4,0) and P3(2,3).

In [4]:
```
from sympy import Point, Triangle
P1 = Point(0, 0)
P2 = Point(4, 0)
P3 = Point(2, 3)
triangle = Triangle(P1, P2, P3)
print("Triangle area:", triangle.area)
print("Triangle perimeter:", triangle.perimeter)
print("Centroid:", triangle.centroid)
```

```
Triangle area: 6
Triangle perimeter: 4 + 2*sqrt(13)
Centroid: Point2D(2, 1)
```

Q4. Write a Python program using the SymPy library to perform geometrical transformations on a straight line in a 2D plane.

In [5]:
```
from sympy import Line, Point, pi
line = Line(Point(1, 1), Point(4, 4))
translated_line = line.translate(2, -1)
rotated_line = line.rotate(pi/2)
print("Original line:", line)
print("Translated line:", translated_line)
print("Rotated line:", rotated_line)
```

```
Original line: Line2D(Point2D(1, 1), Point2D(4, 4))
Translated line: Line2D(Point2D(3, 0), Point2D(6, 3))
Rotated line: Line2D(Point2D(-1, 1), Point2D(-4, 4))
```

Q5. Write a Python program using the SymPy library to perform reflection and scaling transformations on a straight line in a 2D plane.

In [6]:
```
from sympy import Line, Point
line = Line(Point(2, 3), Point(6, 7))
reflected_line = line.reflect(Line(Point(0,0), Point(1,0)))
scaled_line = line.scale(2, 2)
print("Original line:", line)
print("Reflected line:", reflected_line)
print("Scaled line:", scaled_line)
```

```
Original line: Line2D(Point2D(2, 3), Point2D(6, 7))
Reflected line: Line2D(Point2D(2, -3), Point2D(6, -7))
Scaled line: Line2D(Point2D(4, 6), Point2D(12, 14))
```