TRIBHUVAN UNIVERSITY

Institute of Engineering

Pulchowk Campus

A LAB REPORT ON

Assignment 2

LAB NO. :

EXPERIMENT DATE : 2025-08-22

SUBMITTED DATE : 2025-08-25

SUBMITTED BY

Name : Abhishek Kharel

Group : A

Roll No. : 081BEL005

SUBMITTED TO

Department of

Computer Engineering

Assignment 2

Operator overloading for: Arithmetic and relational operator

In [1]:
```python
import math

class Vector2D:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
    def __str__(self):
        return f"({self.x}, {self.y})"
    def __add__(self, other):
        return Vector2D(self.x + other.x, self.y + other.y)
    def __sub__(self, other):
        return Vector2D(self.x - other.x, self.y - other.y)
    def __mul__(self, scalar):
        if isinstance(scalar, (int, float)):
            return Vector2D(self.x * scalar, self.y * scalar)
        raise TypeError("Multiplication only supports a scalar (int or float).")
    def __rmul__(self, scalar):
        return self.__mul__(scalar)
    def __truediv__(self, scalar):
        if scalar == 0:
            raise ValueError("Cannot divide by zero.")
        return Vector2D(self.x / scalar, self.y / scalar)
    def __matmul__(self, other):
        return self.x * other.x + self.y * other.y
    def magnitude(self):
        return math.sqrt(self.x**2 + self.y**2)
    def __eq__(self, other):
        return self.magnitude() == other.magnitude()
    def __ne__(self, other):
        return self.magnitude() != other.magnitude()
    def __lt__(self, other):
        return self.magnitude() < other.magnitude()
    def __le__(self, other):
        return self.magnitude() <= other.magnitude()
    def __gt__(self, other):
        return self.magnitude() > other.magnitude()
    def __ge__(self, other):
        return self.magnitude() >= other.magnitude()
    def direction(self):
        return math.degrees(math.atan2(self.y, self.x))
    def normalize(self):
        mag = self.magnitude()
        if mag == 0:
            return Vector2D(0, 0)
        return Vector2D(self.x / mag, self.y / mag)
    def distance_to(self, other):
        return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)
```

```python
v1 = Vector2D(3, 4)
v2 = Vector2D(1, 2)

print("v1:", v1)
print("v2:", v2)
print("Addition:", v1 + v2)
print("Subtraction:", v1 - v2)
print("v1 * 2:", v1 * 2)
print("2 * v2:", 2 * v2)
print("v1 / 2:", v1 / 2)
print("Dot product:", v1 @ v2)

print("v1 magnitude:", v1.magnitude())
print("v2 magnitude:", v2.magnitude())
print("v1 direction:", v1.direction())
print("v2 direction:", v2.direction())

print("v1 == v2?", v1 == v2)
print("v1 != v2?", v1 != v2)
print("v1 > v2?", v1 > v2)
print("v1 < v2?", v1 < v2)
print("v1 normalized:", v1.normalize())
print("Distance between v1 and v2:", v1.distance_to(v2))
```

```
v1: (3, 4)
v2: (1, 2)
Addition: (4, 6)
Subtraction: (2, 2)
v1 * 2: (6, 8)
2 * v2: (2, 4)
v1 / 2: (1.5, 2.0)
Dot product: 11
v1 magnitude: 5.0
v2 magnitude: 2.23606797749979
v1 direction: 53.13010235415598
v2 direction: 63.43494882292201
v1 == v2? False
v1 != v2? True
v1 > v2? True
v1 < v2? False
v1 normalized: (0.6, 0.8)
Distance between v1 and v2: 2.8284271247461903
```