In [128…

```
#   19p0012 Aitzaz Tahir Ch
#   Numerical computing assignment 01
```

In [129…

```
#Vector class with 2 elements.
class Vector:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def __str__(self):
        return "[" + str(self.length) + "," + str(self.width) + "]"

    def add(self, other):
        #sum of self and other vector.Both must be vectors.
        return Vector(self.length + other.length, self.width + other.width)

    def sub(self, other):
        #subtraction of self and other vector.Both must be vectors.
        return Vector(self.length - other.length, self.width - other.width)

    def mul(self, scalar):
        #multiplication of a vector with a scalar
        if isinstance(scalar, int) or isinstance(scalar, float):
            return Vector(self.length * scalar, self.width * scalar)
        raise NotImplementedError('Vectors can only be multiplied by a scalar')
```

In [130…

```
v1 = Vector(2,3)
v2 = Vector(4,5)
print('vector 1 : ',v1)
print('vector 2 : ',v2)
print('Adding vector2 to vector1 : ',v1.add(v2))
print('Subtracting vector2 from vector1 : ',v1.sub(v2))
print('Multiplying vector1 by 2 : ',v1.mul(2))
```

```
vector 1 :  [2,3]
vector 2 :  [4,5]
Adding vector2 to vector1 :  [6,8]
Subtracting vector2 from vector1 :  [-2,-2]
Multiplying vector1 by 2 :  [4,6]
```

In [131…

```
def dot_product(self, other):
    #dot product of self and other vector.Both must be vectors.
        if not isinstance(other, Vector):
            raise TypeError('Dot product can only be of two Vectors')
        else:
            return self.length * other.length + self.width * other.width
Vector.dot_product = dot_product
```

In [132…

```
print('Dot product of vector1 and vector2 : ',v1.dot_product(v2))
```

```
Dot product of vector1 and vector2 :  23
```

In [133…

```
#Vector class with 3 elements.
class Vector3:
    def __init__(self, length, width, height):
        self.length = length
        self.width = width
        self.height = height

    def __str__(self):
        return "[" + str(self.length) + "," + str(self.width) + "," + str(self.height) + "]"

    def add(self, other):
        #sum of self and other vector.Both must be vectors.
        return Vector3(self.length + other.length, self.width + other.width, self.height + other.height)

    def sub(self, other):
        #subtraction of other vector by self.Both must be vectors.
        return Vector3(self.length - other.length, self.width - other.width, self.height - other.height)

    def mul(self, scalar):
        #multiplication of a vector with a scalar.
        if isinstance(scalar, int) or isinstance(scalar, float):
            return Vector3(self.length * scalar, self.width * scalar, self.height * scalar)
        raise NotImplementedError('Vectors can only be multiplied by a scalar')

    def dot_product(self, other):
    #dot product of self and other vector.Both must be vectors.
        if not isinstance(other, Vector3):
            raise TypeError('Dot product can only be between Vectors')
        else:
            return self.length * other.length + self.width * other.width + self.height * other.height
```

In [134…

```
V1 = Vector3(1,2,3)
V2 = Vector3(4,5,6)
print('vector 1 : ',V1)
print('vector 2 : ',V2)
print('Adding vector2 to vector1 : ',V1.add(V2))
print('Subtracting vector2 from vector1 : ',V1.sub(V2))
print('Multiplying vector1 by 2 : ',V1.mul(2))
print('Dot product of vector1 and vector2 : ',V1.dot_product(V2))
```

```
vector 1 :  [1,2,3]
vector 2 :  [4,5,6]
Adding vector2 to vector1 :  [5,7,9]
Subtracting vector2 from vector1 :  [-3,-3,-3]
Multiplying vector1 by 2 :  [2,4,6]
Dot product of vector1 and vector2 :  32
```

In [135…

```
#Vector class with N elements.
#we can use *args so it can take any number of elements in parameters and we can perform all the operations on it.
class VectorN:
    def __init__(self, *args):
        self.args = args
        self.args = list(self.args)

    def __str__(self):
        return str(self.args)

    def add(self, other):
        #sum of self and other vector.Both must be vectors.
        result = tuple(map(lambda i, j: i + j, self.args, other.args))
        return list(result)

    def sub(self, other):
        #subtraction of other vector by self.Both must be vectors.
        result = tuple(map(lambda i, j: i - j, self.args, other.args))
        return list(result)
    def mul(self, scalar):
        #multiplication of a vector with a scalar.
        if isinstance(scalar, int) or isinstance(scalar, float):
            x = tuple([i * scalar for i in self.args])
            return list(x)
        raise NotImplementedError('Vectors can only be multiplied by a scalar')

    def dot_product(self, other):
    #dot product of self and other vector.Both must be vectors.
        if not isinstance(other, VectorN):
            raise TypeError('Dot product can only be between Vectors')
        else:
            b = 0
            result = tuple(map(lambda i, j: i * j, self.args, other.args))
            for a in result:
                b = b+a
            return b
```

In [136…

```
Vn1 = VectorN(1,2,3,4,5,6)
Vn2 = VectorN(1,2,3,4,5,6)
print('vector 1 : ',Vn1)
print('vector 2 : ',Vn2)
print('Adding vector2 to vector1 : ',Vn1.add(Vn2))
print('Subtracting vector2 from vector1 : ',Vn1.sub(Vn2))
print('Multiplying vector1 by 2 : ',Vn1.mul(2))
print('Dot product of vector1 and vector2 : ',Vn1.dot_product(Vn2))
```

```
vector 1 :  [1, 2, 3, 4, 5, 6]
vector 2 :  [1, 2, 3, 4, 5, 6]
Adding vector2 to vector1 :  [2, 4, 6, 8, 10, 12]
Subtracting vector2 from vector1 :  [0, 0, 0, 0, 0, 0]
Multiplying vector1 by 2 :  [2, 4, 6, 8, 10, 12]
Dot product of vector1 and vector2 :  91
```

In [ ]:

In [ ]: