# EX - 3 VECTOR

**T SADAKOPA RAMAKRISHNAN | IT - B | 3122225002109**

## AIM:

To create a Vector class that can create a vector object on which arithmetic operations such as addition, subtraction, scalar multiplication can be performed. The Vector class creates a vector object of a sequence if a sequence is passed, or creates a Vector object of a certain input length consisting of 0s.

## ALGORITHM:

Step 1: Start the Algorithm
Step 2: Create a class Vector and Inside the class, define an init method that takes a single argument, val.
Step 3: In the init method, use an if-elif-else block to check if the type of val is either an integer or a list.
Step 4: If val is an integer, set the instance variable dimension to val, and set the instance variable list to a list of zeros with length val.
Step 5: If val is a list, set the instance variable dimension to the length of the list, and set the instance variable list to the list.
Step 6: If val is neither an integer nor a list, raise a TypeError with the message "Enter only an integer or list."
Step 7: Define a len method that returns the instance variable dimension.
Step 8: Define a getitem method that takes an index as an argument and returns the value of the list at that index.
Step 9: Define a setitem method that takes an index and a value as arguments and sets the value of the list at that index to the given value.
Step 10: Define a add method that takes another Vector object as an argument and returns a new Vector object whose values are the sum of the corresponding values of the two vectors. Raise a ValueError if the dimensions of the two vectors are different.
Step 11: Define a sub method that takes another Vector object as an argument and returns a new Vector object whose values are the difference of the corresponding values of the two vectors. Raise a ValueError if the dimensions of the two vectors are different.
Step 12: Define a mul method that takes another Vector object as an argument and returns a new Vector object whose values are the product of the corresponding values of the two vectors. Raise a ValueError if the dimensions of the two vectors are different.
Step 13: Define a truediv method that takes another Vector object as an argument and returns a new Vector object whose values are the quotient of the corresponding values

of the two vectors. Raise a ValueError if the dimensions of the two vectors are different. Raise a ZeroDivisionError if any of the elements of the other Vector object are 0.
Step 14: Define a str method that returns the string representation of the list instance variable.
Step 15: End the Algorithm

## CODE:

```
'''This module creates vectors of vector class and
performs addition, subtraction, multiplication of two
vectors.This is a part of the exercises given
 under the course UIT2201 (Programming
and Data Structures).

In this source code I've executed my own logic and may contain
bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Wed Apr 19 2023

Revised on Wed Apr 22 2023

Original Author: T. Sadakopa Ramakrishnan
<sadakopa2210221@ssn.edu.in>
'''


class Vector:

  def __init__(self, val):
    if isinstance(val, int):
      self.dimension = val
      self.list = [0 for i in range(val)]
    elif isinstance(val, list):
      self.dimension = len(val)
      self.list = val
    else:
```

```
    raise TypeError("Enter only an integer or list.")

def __len__(self):
  return self.dimension

def __getitem__(self, index):
  return self.list[index]

def __setitem__(self, index, val):
  self.list[index] = val
  return self.list

def __add__(self, other):
  if self.dimension != len(other):
    raise ValueError("Dimensions of two lists do not match")
  else:
    result = [0 for i in range(self.dimension)]
    for i in range(self.dimension):
      result[i] = self.list[i] + other.list[i]
    return result

def __sub__(self, other):
  if self.dimension != len(other):
    raise ValueError("Dimensions of two lists do not match")
  else:
    result = [0 for i in range(self.dimension)]
    for i in range(self.dimension):
      result[i] = self.list[i] - other.list[i]
    return result

def __mul__(self, other):
  if self.dimension != len(other):
    raise ValueError("Dimensions of two lists do not match")
  else:
    result = [0 for i in range(self.dimension)]
    for i in range(self.dimension):
      result[i] = self.list[i] * other.list[i]
    return result

def __truediv__(self, other):
  if self.dimension != other.dimension:
```

```
      raise ValueError("Dimensions of two lists do not match")
    else:
      result = [0] * self.dimension
      for iteration in range(self.dimension):
        if other.list[iteration] == 0:
          raise ZeroDivisionError("Cannot divide with 0")
        else:
          result[iteration] = self.list[iteration] /
other.list[iteration]
      return result

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


if __name__ == "__main__":
    #Creating an empty Vector
    v = Vector(5)
    print("Vector V: ", v)
    print()

    #Assigning two values to the vector
    v[0] = 1
    v[4] = 2
    print("New vector V: ", v)
    print()

    #Creating a new vector V1
    v1 = Vector([1, 2, 3, 4])
    print("V1 Vector: ", v1)
    print()

    #Printing out the value in index 2
    print("Second index in V1: ", v1[2])
    print()

    #Assigning a value to 0th index
    v1[0] = 5
    print("New V1 vector: ", v1)
    print()
```

```
#Creating a new vector V2
v2 = Vector([1, 2, 3, 4])
print("V2: ", v2)
print()

#Adding two vectors
print("v1+v2:", v1 + v2)
print()

#Subtracting two vectors
print("v1 - v2: ", v1 - v2)
print()

#Multiplying two vectors
print("v1*v2: ", v1 * v2)
print()

#Dividing two vectors
print("v1 / v2:", v1 / v2)
print()
```

## OUTPUT:

```
Vector V:  [0, 0, 0, 0, 0]
New vector V:  [1, 0, 0, 0, 2]
V1 Vector:  [1, 2, 3, 4]
Second index in V1:  3
New V1 vector:  [5, 2, 3, 4]
V2:  [1, 2, 3, 4]
v1+v2: [6, 4, 6, 8]
v1 - v2:  [4, 0, 0, 0]
v1*v2:  [5, 4, 9, 16]
v1 / v2: [5.0, 1.0, 1.0, 1.0]
```