Programmer un type vecteur

```
#Ajouter une instruction permettant d’utiliser les fonctions usuelles

class t_vecteur:

    def __init__(self, coordonnees):
        """Constructeur qui place dans l'attribut de <self> les_coordonnees le tuple
contenant les valeurs de <coordonnees>"""

    def __del__(self):
        """Détruit l'attribut <les_coordonnees> de <self>"""

    def __str__(self):
        """Converti <self> en une chaine de caractères pour l'affichage pour une
utilisateur humain."""

    def __getitem__(self, key):
        """retourne la composante d'indice <key> dans le vecteur <self>"""

    def dimension(self):
        """Fournit le nombre de coordonnées du vecteur <self>"""

    def __add__(self, other):
        """retorune la somme de <self> et <other>"""

    def __mul__(self, coefficient):
        """retorune <coefficient> fois le vecteur <self> (sic l'ordre)"""

    def __matmul__(self, other):
        """retorune le produit scalaire de <self> et de <other>"""

    def norme_carre(self):
        """retorune le carré de la norme de <self>"""

    def norme(self):
        """retorune la norme de <self>"""

    def normalise(self):
        """retourne un vecteur ayant même direction et même sens que <self> mais dont la
norme vaut 1."""
```

Exemple de test unitaire

```
if __name__ == "__main__":

    mes_vecteurs = []
    mes_vecteurs.append(t_vecteur([1, 2, -3]))
    mes_vecteurs.append(t_vecteur([7, 0, -3]))
    mes_vecteurs.append(t_vecteur([0, 4, 5]))

    for v in mes_vecteurs:
        print(v)
```

```
    print(mes_vecteurs[1] + mes_vecteurs[0])
    print(mes_vecteurs[2] * 2)
    print(mes_vecteurs[0] @ mes_vecteurs[1])
    print(mes_vecteurs[2].norme())
    print(mes_vecteurs[2].normalise())
```

Affichage attendu

(1, 2, -3)

(7, 0, -3)

(0, 4, 5)

(8, 2, -6)

(0, 8, 10)

16

6.4031242374328485

(0.0, 0.6246950475544243, 0.7808688094430304)