Fiche 1 - Python

Géométrie dans l'espace

En Python, on exprimera les coordonnées d'un point ou d'un vecteur par une liste de 3 réels.

1. Ecrire la fonction `somme_vecteur` qui prend en argument deux listes de 3 éléments et renvoie la somme vectoriel.

   Exemple :

```
>>> somme_vecteur([2,1,6],[8,-2,3])
[10, -1, 9]
```

2. Ecrire la fonction `multiplication_scalaire` qui prend en argument un réel k et un vecteur $\vec{u}$ et renvoie le produit scalaire $k\vec{u}$.

   Exemple :

```
>>> multiplication_scalaire ( 3 , [2,1,6])
[6, 3, 18]
```

3. Ecrire la fonction `point_droite` qui prend en argument un point A ,un vecteur $\vec{u}$ et un réel t et renvoie le point de la droite $(A, \vec{u})$ lié au paramètre t. Autrement dit, pour la représentation de la droite passant par $A(x_A, y_A, z_A)$ et de direction $\vec{u}\begin{pmatrix} a \\ b \\ c \end{pmatrix}$:

$$\begin{cases} x &= x_A + a \times t \\ y &= y_A + b \times t \\ z &= z_A + c \times t \end{cases} \quad , t \in \mathbb{R}$$

   La fonction renvoie le point du paramètre t.

   Exemple :

```
>>> point_droite([3,-5,2] , [2,3,1],1)
[5, -2, 3]
```

4. Ecrire la fonction `appartient` qui prend en argument un point M, un point A ,un vecteur $\vec{u}$ et renvoie le booléen correspondant à l'appartenance du point M à la droite $(A, \vec{u})$.

   Exemple :

```
>>> appartient([5, -2, 3] , [3,-5,2] , [2,3,1])
True
>>> appartient([4, -2, 3] , [3,-5,2] , [2,3,1])
False
```

Fiche 1 - Python
Correction

En Python, on exprimera les coordonnées d'un point ou d'un vecteur par une liste de 3 réels.

1. Ecrire la fonction `somme_vecteur` qui prend en argument deux listes de 3 éléments et renvoie la somme vectoriel.

   Exemple :

```
>>> somme_vecteur([2,1,6],[8,-2,3])
[10, -1, 9]
```

   Correction

```
def somme_vecteur (u,v) :
    return [ u[0]+v[0] , u[1]+v[1] , u[2]+v[2]]
```

2. Ecrire la fonction `multiplication_scalaire` qui prend en argument un réel k et un vecteur $\vec{u}$ et renvoie le produit scalaire $k\vec{u}$.

   Exemple :

```
>>> multiplication_scalaire ( 3 , [2,1,6])
[6, 3, 18]
```

   Correction

```
def multiplication_scalaire ( k , u ) :
    return [ k * u[0] , k *u[1] , k* u[2]]
```

3. Ecrire la fonction `point_droite` qui prend en argument un point A ,un vecteur $\vec{u}$ et un réel t et renvoie le point de la droite $(A,\vec{u})$ lié au paramètre t. Autrement dit, pour la représentation de la droite passant par $A(x_A,y_A,z_A)$ et de direction $\vec{u}\begin{pmatrix}a\\b\\c\end{pmatrix}$:

$$\begin{cases} x &= x_A + a\times t \\ y &= y_A + b\times t \\ z &= z_A + c\times t \end{cases} \quad , t\in\mathbb{R}$$

   La fonction renvoie le point du paramètre t.

   Exemple :

```
>>> point_droite([3,-5,2] , [2,3,1],1)
[5, -2, 3]
```

   Correction

```
def point_droite( a , u , t ) :
    return somme_vecteur( a , multiplication_scalaire(t,u))
```

4. Ecrire la fonction `appartient` qui prend en argument un point M, un point A ,un vecteur $\vec{u}$ et renvoie le booléen correspondant à l'appartenance du point M à la droite $(A, \vec{u})$.

   Exemple :

```
>>> appartient([5, -2, 3] , [3,-5,2] , [2,3,1])
True
>>> appartient([4, -2, 3] , [3,-5,2] , [2,3,1])
False
```

Correction

```
def appartient( m , a , u ) :
    [x,y,z] = m
    [x_a , y_a , z_a ] = a
    [ x_u , y_u , z_u ] = u
    t_x = (x - x_a)/x_u
    t_y = (y - y_a)/y_u
    t_z = (z - z_a)/z_u
    return (t_x == t_y) and (t_y == t_z)
```