TP10 En Python

Vecteurs égaux - Vecteurs opposés

Allumer l'ordinateur et connectez-vous en utilisant votre login et votre mot de passe puis lancer « Python en ligne : https://www.codabrainy.com/python-compiler/ ».

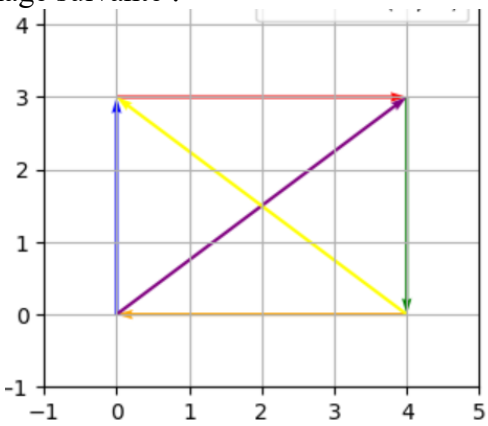
Exercice : Représentation graphique et égalité de vecteurs en Python

L'objectif est d'analyser les vecteurs du plan, leur représentation graphique et les notions de vecteurs égaux et de vecteurs opposés, sans tenir compte de leur point d'application.

On considère le script Python ci-dessous utilisant les bibliothèques **NumPy** et **Matplotlib**. Il permet de tracer plusieurs vecteurs du plan à partir de différentes origines.
Les vecteurs sont définis par leurs coordonnées, puis représentés graphiquement à l'aide de la fonction `quiver`.

Objectifs de l'exercice

Cet exercice vise à representer l'image suivante :



- comprendre la représentation graphique d'un vecteur dans le plan ;
- observer que deux vecteurs peuvent être **égaux** même s'ils n'ont pas la même origine

Première question : Compléter l'algorithme suivant en utilisant les notions et résultats du TP 9.

```
import numpy as np
import matplotlib.pyplot as plt

# Définition des vecteurs
v1 = np.array([4, 0])
v2 = …….....
v3 = ……..
v4 = ……..

# Origine commune
origin1 = np.array([0, 3])
origin2 = ……...
origin3 = ……...
origin4 = ……...
```

```
# Tracé
plt.quiver(*origin1, *v1, angles='xy', scale_units='xy', scale=1, color='red', label='v1 = (4, 0)')
plt.quiver(……..)
plt.quiver(……..)
plt.quiver(……..)

# Mise en forme
plt.xlim(-1, 5)
plt.ylim(-1, 7)
plt.gca().set_aspect('equal')
plt.grid()
plt.legend()
plt.title("Construire graphiquement un rectangle à partir de vecteurs orientés.")

plt.show()
```

Deuxième question :

Représenter les deux vecteurs diagonaux v5 et v6 en suivant le modèle de l'image ci-dessus.

Correction TP10

```
import numpy as np
import matplotlib.pyplot as plt

# Définition des vecteurs
v1 = np.array([4, 0])
v2 = np.array([0, 3])
v3 = np.array([-4, 0])
v4 = np.array([0, -3])
v5 = np.array([4, 3])
v6 = np.array([-4, 3])

# Origine commune
origin1 = np.array([0, 3])
origin2 = np.array([0, 0])
origin3 = np.array([4, 0])
origin4 = np.array([4, 3])
origin5 = np.array([0, 0])
origin6 = np.array([4, 0])

# Tracé
plt.quiver(*origin1, *v1, angles='xy', scale_units='xy', scale=1, color='red', label='v1 = (4, 0)')
plt.quiver(*origin2, *v2, angles='xy', scale_units='xy', scale=1, color='blue', label='v2 = (0, 3)')
plt.quiver(*origin3, *v3, angles='xy', scale_units='xy', scale=1, color='orange', label='v3 = (-4, 0)')
plt.quiver(*origin4, *v4, angles='xy', scale_units='xy', scale=1, color='green', label='V4 = (0, -3)')
plt.quiver(*origin5, *v5, angles='xy', scale_units='xy', scale=1, color='purple', label='V5 = (4, 3)')
plt.quiver(*origin6, *v6, angles='xy', scale_units='xy', scale=1, color='yellow', label='V6 = (-4, -3)')


# Mise en forme
plt.xlim(-1, 5)
plt.ylim(-1, 7)
plt.gca().set_aspect('equal')
plt.grid()
plt.legend()
plt.title("Construire graphiquement un rectangle à partir de vecteurs orientés.")

plt.show()
```