TP6 En Python

Représentation d'un parallélogramme dans un repère orthonormé centré et gradué à l'aide de Python

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/ »

Activité N°1

L'objectif est de concevoir un **algorithme Python** capable de :

- Définir les sommets d'un parallélogramme à partir de trois points connus.
- Calculer automatiquement le quatrième sommet pour garantir la condition de parallélisme.
- Tracer le parallélogramme dans un **repère orthonormé centré à l'origine**, avec **graduations visibles uniquement sur les axes Ox et Oy**, comme sur un graphique de géométrie analytique.

```
#------------------------------ Activité N°1----------------------------------------
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(7,7))

# Position des axes au centre
ax.spines['left'].set_position('zero')   # Axe des ordonnées (y)
ax.spines['bottom'].set_position('zero') # Axe des abscisses (x)
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')

# Graduations sur les axes uniquement
ax.set_xticks(range(-6, 7))
ax.set_yticks(range(-6, 7))
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')

# Grille et mise en forme
ax.grid(True, which='both', color='gray', linestyle=':', linewidth=0.6)
ax.set_xlim(-6, 6)
ax.set_ylim(-6, 6)
ax.set_aspect('equal', adjustable='box')
ax.set_title("Parallélogramme dans un repère orthonormé centré et gradué", pad=15)

# Points connus
A = (-2, -1)
B = (3, -1)
D = (-1, 3)

# Calcul du quatrième point
C = (B[0] + D[0] - A[0], B[1] + D[1] - A[1])

# Coordonnées pour le tracé
x = [A[0], B[0], C[0], D[0], A[0]]
y = [A[1], B[1], C[1], D[1], A[1]]

# Tracé
ax.plot(x, y, 'bo-', linewidth=2)
ax.fill(x, y, alpha=0.1, color='blue')

# Étiquettes des sommets
ax.text(A[0]-0.3, A[1]-0.3, 'A', fontsize=12)
ax.text(B[0]+0.2, B[1]-0.3, 'B', fontsize=12)
ax.text(C[0]+0.2, C[1]+0.2, 'C', fontsize=12)
ax.text(D[0]-0.3, D[1]+0.2, 'D', fontsize=12)

plt.show()
```

Résultats :

A(;), B(;), C(;), D(;),

Activité N°2

On équipe maintenant ton travail d'un **algorithme qui :**

- trace le parallélogramme,
- trace ses deux diagonales,
- **calcule et affiche** le point d'intersection de ces diagonales (théoriquement le centre du parallélogramme).

Au dessous une version complète en Python, dans le même style que ce que tu as déjà.

```
#------------------------------ Activité N°2----------------------------------------
# --- 4) Tracé des deux diagonales AC et BD ---
ax.plot([A[0], C[0]], [A[1], C[1]], linestyle='--')  # diagonale AC
ax.plot([B[0], D[0]], [B[1], D[1]], linestyle='--')  # diagonale BD

# --- 5) Calcul du point d'intersection des diagonales ---
Ix = (A[0] + C[0]) / 2
Iy = (A[1] + C[1]) / 2
I = (Ix, Iy)

# --- 6) Représentation du point d'intersection ---
ax.plot(Ix, Iy, 'ro', markersize=6)  # point rouge
ax.text(Ix + 0.2, Iy + 0.2, f"I({Ix:.1f} ; {Iy:.1f})", fontsize=11, color='red')

# (optionnel) Affichage des coordonnées dans la console
print("Coordonnées du point d'intersection des diagonales I :", I)

plt.show()
```

Coordonnées du point d'intersection des diagonales I : (;)

Ce que fait cet ajout

- Trace les **deux diagonales** [AC] et [BD] en pointillés.
- Calcule le **point d'intersection I** comme milieu de [AC].
- Affiche **graphiquement** ce point en rouge avec ses coordonnées.
- Affiche aussi les coordonnées dans la **console Python**.

Activité N°3

Très bon réflexe ! Ajouter la **représentation graphique colorée** des deux parallélogrammes — l'original et celui tourné de 90° — rend le travail bien plus visuel et complet.

Voici donc un **programme Python complet** qui :

1. Crée le repère orthonormé centré et gradué,
2. Trace le **parallélogramme initial** en **bleu**,
3. Calcule la rotation de **90°** (anti-horaire) autour de l'origine,
4. Trace le **parallélogramme tourné** en **rouge**,
5. Affiche aussi le **point d'intersection des diagonales avant et après rotation**.

```
#------------------------------ Activité N°3----------------------------------------
# --- 2) Fonction de rotation de 90° autour de l'origine ---
def rotation_90(point):
    x, y = point
    return (-y, x)

# Points tournés
A2 = rotation_90(A)
B2 = rotation_90(B)
C2 = rotation_90(C)
D2 = rotation_90(D)
I2 = rotation_90(I)

# --- 5) Tracé du parallélogramme tourné (rouge) ---
x2 = [A2[0], B2[0], C2[0], D2[0], A2[0]]
y2 = [A2[1], B2[1], C2[1], D2[1], A2[1]]
ax.plot(x2, y2, 'ro-', linewidth=2, label="Parallélogramme tourné à 90°")
ax.fill(x2, y2, alpha=0.1, color='red')

# Sommets étiquetés
ax.text(A2[0]-0.3, A2[1]-0.3, "A'", fontsize=12, color='red')
ax.text(B2[0]+0.2, B2[1]-0.3, "B'", fontsize=12, color='red')
ax.text(C2[0]+0.2, C2[1]+0.2, "C'", fontsize=12, color='red')
ax.text(D2[0]-0.3, D2[1]+0.2, "D'", fontsize=12, color='red')

# Point I' (centre après rotation)
ax.plot(I2[0], I2[1], 'ro', markersize=6)
ax.text(I2[0]+0.2, I2[1]+0.2, f"I'({I2[0]};{I2[1]})", color='red', fontsize=11)

# --- 6) Légende et affichage ---
ax.legend(loc='upper right')
plt.show()

# --- 7) Affichage des coordonnées dans la console ---
print("Coordonnées après rotation de 90° :")
print("A' =", A2)
print("B' =", B2)
print("C' =", C2)
print("D' =", D2)
print("I' =", I2)
```

Résultats :

A'(…. ; ….) B'(…. ; ….) C'(…. ; ….) D'(…. ; ….) et I'(…. ; ….)