Importer les librairies nécessaires

In [110…

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

Création du jeu de données (6 points)

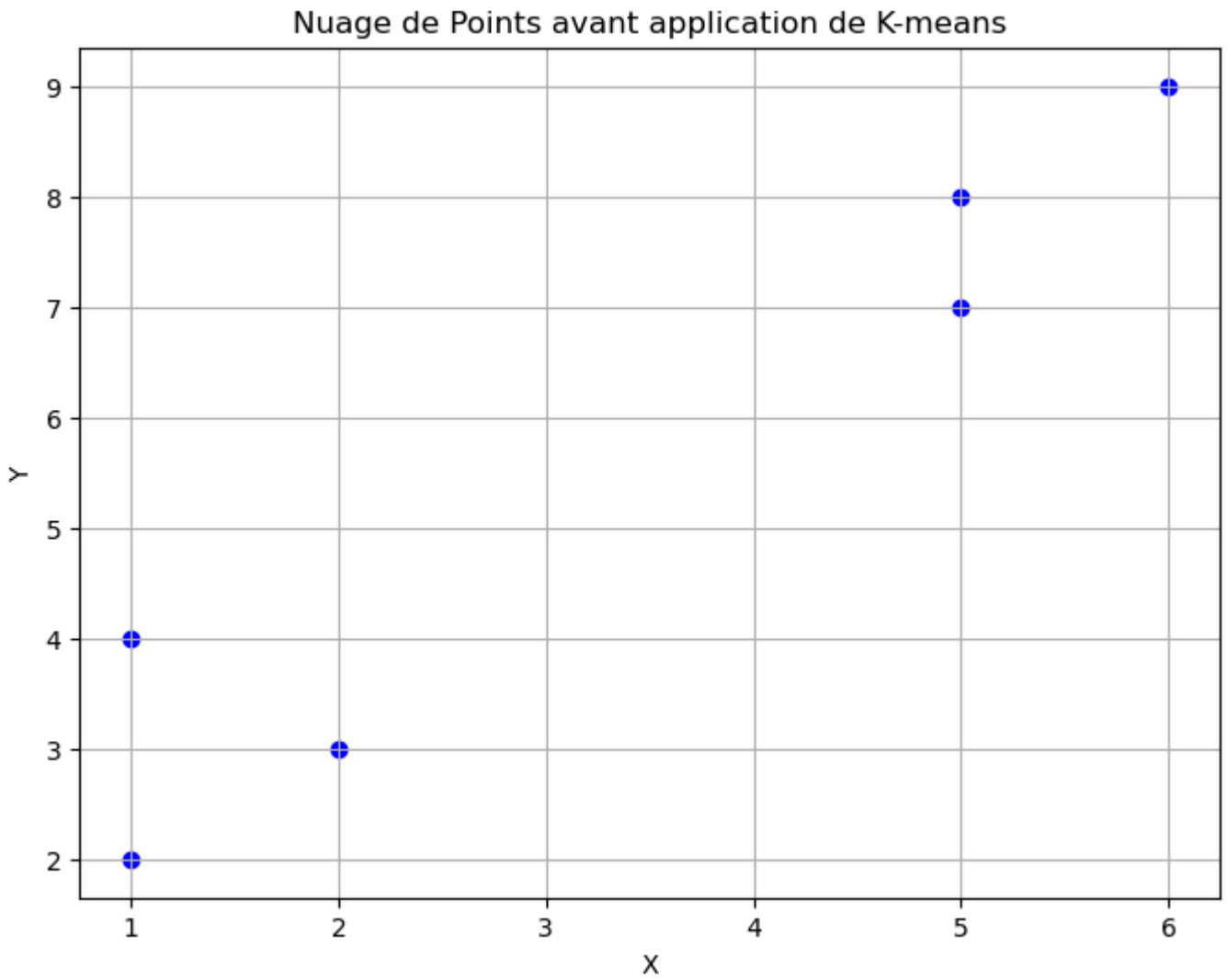
In [113…

```
data = np.array([
    [1, 2],
    [1, 4],
    [2, 3],
    [5, 8],
    [6, 9],
    [5, 7]
])
df = pd.DataFrame(data,columns=['X','Y'])
```

Tracer le nuage de points avant l'application de K-Means

In [116…

```
# Création d'une figure pour le graphique
plt.figure(figsize=(8, 6))  # Définit la taille de la figure (8x6 pouces)

# Création du nuage de points (scatter plot)
plt.scatter(df['X'], df['Y'], color='b', marker='o')  # Trace les points avec les coordonnées X et Y

# Personnalisation du graphique
plt.title('Nuage de Points avant application de K-means')  # Ajoute un titre au graphique
plt.xlabel('X')  # Ajoute une étiquette pour l'axe des X
plt.ylabel('Y')  # Ajoute une étiquette pour l'axe des Y
plt.grid(True)  # Affiche une grille pour mieux visualiser les points
plt.show()  # Affiche le graphique à l'écran
```



Définition des centroïdes initiaux (1er et 4ème point)

In [119…

```
# Initialisation des centroïdes à partir des données existantes
initial_centroids = np.array([
    df.iloc[0],  # Le premier point du DataFrame, qui correspond aux coordonnées (1, 2)
    df.iloc[3]   # Le quatrième point du DataFrame, qui correspond aux coordonnées (5, 8)
```

```
])

# Affichage des centroïdes initialisés
print("Centroïdes initiaux :")
print(initial_centroids)
```

```
Centroïdes initiaux :
[[1 2]
 [5 8]]
```

Application de K-Means avec ces centroïdes initiaux

In [122…

```
# Application de l'algorithme K-Means avec des paramètres spécifiques
kmeans = KMeans(
    n_clusters=2,            # Nombre de clusters souhaités, ici 2 clusters
    init=initial_centroids,  # Initialisation des centroïdes, ici avec des points spécifiques (définis précédemment)
    max_iter=100,            # Le nombre maximum d'itérations pour l'algorithme, ici limité à 100 itérations
    n_init=1,                # Le nombre d'initialisations différentes des centroïdes. Ici, une seule initialisation est effec
    random_state=42          # Fixe la valeur de la graine pour l'aléatoire, garantissant que les résultats sont reproductible
)
kmeans.fit(df[['X','Y']])
#nombre d'iteration avant la convergence
iteration=kmeans.n_iter_
#récuperation des centroides finaux
final_centroides=kmeans.cluster_centers_
```

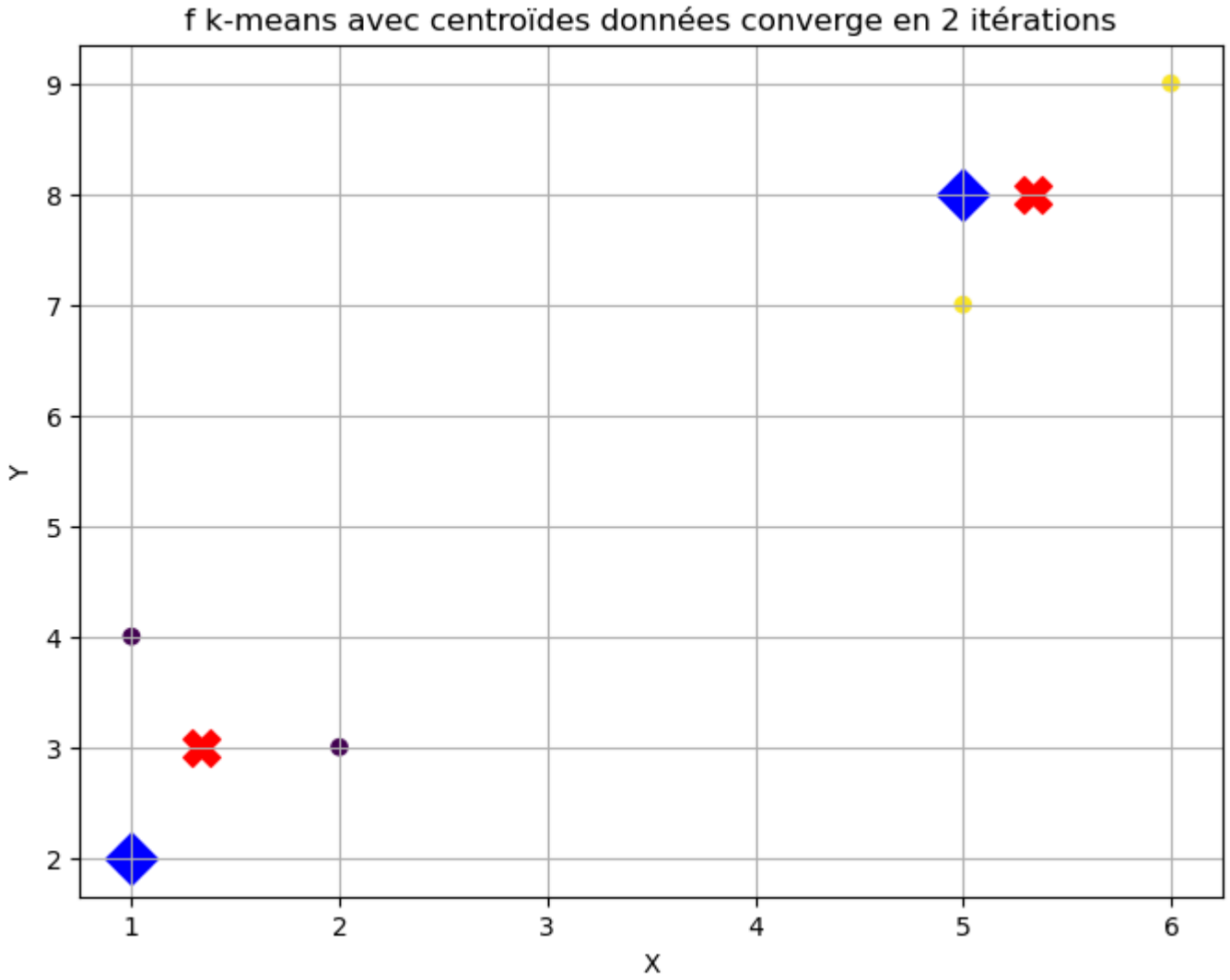
```
C:\Users\tb_ra\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
P_NUM_THREADS=1.
  warnings.warn(
```

affichage des résultats

In [125…

```
# Création d'une figure pour le graphique
plt.figure(figsize=(8, 6))  # Définit la taille du graphique (8x6 pouces)

# Création du nuage de points (scatter plot)
# Trace les points avec les coordonnées X et Y
# `c=kmeans.labels_` colore les points selon leur cluster, avec une palette 'viridis'
```

```
# `label="points"` ajoute une légende pour les points
plt.scatter(df['X'], df['Y'], c=kmeans.labels_, cmap='viridis', label="points")

# Affichage des centroïdes initiaux
# `initial_centroids[:, 0]` et `initial_centroids[:, 1]` récupèrent les coordonnées X et Y des centroïdes initiaux
# `c='blue'` et `marker='D'` définissent respectivement la couleur et le type de marqueur
# `s=200` ajuste la taille du marqueur
plt.scatter(initial_centroids[:, 0], initial_centroids[:, 1], c='blue', marker='D', s=200, label="centroïdes initiaux")

# Affichage des centroïdes finaux
# `final_centroides[:, 0]` et `final_centroides[:, 1]` récupèrent les coordonnées X et Y des centroïdes finaux
# `c='red'` et `marker='X'` définissent respectivement la couleur et le type de marqueur
plt.scatter(final_centroides[:, 0], final_centroides[:, 1], c='red', marker='X', s=200, label="centroïdes finaux")


# Personnalisation du graphique
plt.title(f"f k-means avec centroïdes données converge en {iteration} itérations")  # Affiche le nombre d'itérations
plt.xlabel('X')  # Ajoute une étiquette pour l'axe des X
plt.ylabel('Y')  # Ajoute une étiquette pour l'axe des Y
plt.grid(True)  # Affiche une grille pour mieux visualiser les points
plt.show()  # Affiche le graphique à l'écran
```


f k-means avec centroïdes données converge en 2 itérations
Y
X
1
2
3
4
5
6
2
3
4
5
6
7
8
9

In [ ]:

In [ ]:

In [ ]:

In [ ]: