Atelier_5_corr

N'oubliez pas d'utiliser des commentaires pour expliquer le fonctionnement de chaque partie de votre code.

Activité 1

Générez un tableau NumPy appelé donnees contenant 100 nombres aléatoires entiers entre 1 et 100.

Calculez et affichez la moyenne des valeurs dans le tableau donnees.

Calculez et affichez la médiane des valeurs dans le tableau donnees.

Calculez et affichez l'écart-type des valeurs dans le tableau donnees.

Créez un tableau appelé donnees_pairs qui contient uniquement les valeurs paires de donnees.

Créez un tableau appelé donnees_impaires qui contient uniquement les valeurs impaires de donnees.

Trouvez et affichez la valeur maximale et sa position (indice) dans le tableau donnees.

Triez le tableau donnees par ordre croissant et affichez le résultat.

Vous pouvez utliser les fonctions: numpy.random.randint pour générer des nombres aléatoires entiers. et les fonctions statistiques telles que mean, median, std, et les méthodes comme max et argsort

correction

```
import numpy as np


# Génération du tableau donnees contenant 100 nombres aléatoires entiers entre 1 et 100

donnees = np.random.randint(1, 101, 100)


# Calcul et affichage de la moyenne des valeurs dans le tableau donnees

moyenne = np.mean(donnees)

print("Moyenne:", moyenne)


# Calcul et affichage de la médiane des valeurs dans le tableau donnees

median = np.median(donnees)
```

```
print("Médiane:", median)

# Calcul et affichage de l'écart-type des valeurs dans le tableau donnees
ecart_type = np.std(donnees)
print("Écart-type:", ecart_type)

# Création d'un tableau donnees_pairs contenant uniquement les valeurs paires de donnees
donnees_pairs = donnees[donnees % 2 == 0]

# Création d'un tableau donnees_impaires contenant uniquement les valeurs impaires de donnees
donnees_impaires = donnees[donnees % 2 != 0]

# Affichage des tableaux donnees_pairs et donnees_impaires
print("Données paires:", donnees_pairs)
print("Données impaires:", donnees_impaires)

# Trouver et afficher la valeur maximale et sa position (indice) dans le tableau donnees
max_value = np.max(donnees)
max_index = np.argmax(donnees)
print("Valeur maximale:", max_value, "à la position (indice):", max_index)

# Tri du tableau donnees par ordre croissant et affichage du résultat
donnees_triees = np.sort(donnees)
print("Données triées par ordre croissant:", donnees_triees)
```

Activité 2

Générez deux tableaux NumPy :

notes_maths contenant 20 notes aléatoires entre 10 et 20.

notes_physique contenant 20 notes aléatoires entre 12 et 18.

Calculez la moyenne, l'écart-type et la médiane de chaque ensemble de notes. Affichez les résultats.

Créez un tableau combiné notes_combined en concaténant les deux tableaux précédents.

Trouvez et affichez les indices des éléments dans le tableau combiné qui sont supérieurs à 15.

Remplacez toutes les notes inférieures à 14 dans le tableau combiné par la valeur 14.

Créez un tableau booléen mentions qui indique si chaque note dans le tableau combiné est supérieure ou égale à 16.

Utilisez le tableau mentions pour créer un nouveau tableau etudiants_meritants contenant les notes des étudiants ayant une mention (supérieure ou égale à 16).

Affichez le pourcentage d'étudiants ayant une mention.

Enregistrez le tableau notes_combined dans un fichier CSV avec deux colonnes : "Maths" et "Physique".

Correction

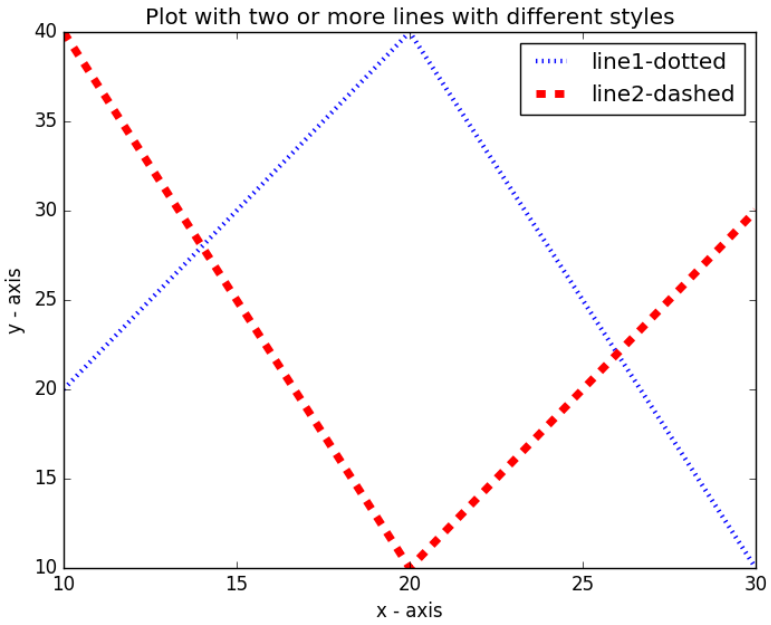
```
import numpy as np


# Génération des deux tableaux NumPy

notes_maths = np.random.uniform(10, 20, 20)

notes_physique = np.random.uniform(12, 18, 20)


# Calcul de la moyenne, l'écart-type et la médiane pour chaque ensemble de notes

moyenne_maths, ecart_type_maths, median_maths = np.mean(notes_maths), np.std(notes_maths), np.median(notes_maths)

moyenne_physique, ecart_type_physique, median_physique = np.mean(notes_physique), np.std(notes_physique), np.median(notes_physique)


# Affichage des résultats

print("Notes Maths - Moyenne:", moyenne_maths, "Écart-type:", ecart_type_maths, "Médiane:", median_maths)

print("Notes Physique - Moyenne:", moyenne_physique, "Écart-type:", ecart_type_physique, "Médiane:", median_physique)
```

```
# Création du tableau combiné notes_combined
notes_combined = np.concatenate([notes_maths, notes_physique])

# Affichage des indices des éléments supérieurs à 15 dans le tableau combiné
indices_sup_15 = np.where(notes_combined > 15)
print("Indices des éléments supérieurs à 15 dans notes_combined:", indices_sup_15)

# Remplacement des notes inférieures à 14 par la valeur 14 dans notes_combined
notes_combined[notes_combined < 14] = 14

# Création du tableau booléen mentions
mentions = notes_combined >= 16

# Création du tableau etudiants_meritants contenant les notes des étudiants avec mention
etudiants_meritants = notes_combined[mentions]

# Affichage du pourcentage d'étudiants avec mention
pourcentage_mention = (np.sum(mentions) / len(mentions)) * 100
print("Pourcentage d'étudiants avec mention:", pourcentage_mention, "%")

# Enregistrement du tableau notes_combined dans un fichier CSV
import pandas as pd

data = {"Maths": notes_maths, "Physique": notes_physique}
df = pd.DataFrame(data)
df.to_csv("notes_combined.csv", index=False)
```

Activité 3

Écrivez un programme Python pour tracer deux lignes ou plus avec des styles différents.

L'extrait de code donne la sortie montrée dans la capture d'écran suivante :



Correction :

```
#Application matplotlib

# line 1 points

x1 = [10,20,30]

y1 = [20,40,10]

# line 2 points

x2 = [10,20,30]

y2 = [40,10,30]

# Set the x axis label of the current axis.

plt.xlabel('x - axis')

# Set the y axis label of the current axis.

plt.ylabel('y - axis')

# Plot lines and/or markers to the Axes.
```

plt.plot(x1,y1, color='blue', linewidth = 3, label = 'line1-dotted',linestyle='dotted')

plt.plot(x2,y2, color='red', linewidth = 5, label = 'line2-dashed', linestyle='dashed')

Set a title

plt.title("Plot with two or more lines with different styles")

show a legend on the plot

plt.legend()

function to show the plot

plt.show()

Activité 4 (tkinter)

Créer une application graphique avec Tkinter qui affiche un bouton et un compteur. Chaque fois que l'utilisateur clique sur le bouton, le compteur est incrémenté et la nouvelle valeur est affichée.
Instructions :
Créez une fenêtre Tkinter avec un titre approprié.
Ajoutez un label affichant le compteur initialisé à 0.
Ajoutez un bouton avec le texte "Cliquez ici".
Définissez une fonction qui sera appelée à chaque clic sur le bouton. Cette fonction doit incrémenter le compteur et mettre à jour le texte du label.
Lancez la boucle principale de l'application.

Correction:

```
import tkinter as tk

class CounterApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Compteur de Clics")

        self.counter = 0

        self.label = tk.Label(self.root, text="0")
        self.label.pack()

        self.button = tk.Button(self.root, text="Cliquez ici", com-
mand=self.increment_counter)
        self.button.pack()
```

```
    def increment_counter(self):
        self.counter += 1
        self.label.config(text=str(self.counter))

def main():
    root = tk.Tk()
    app = CounterApp(root)
    root.mainloop()

if __name__ == "__main__":
    main()
```

Activité 5 (POO et tkinter)

Projet: Carnet d'Adresses

Créez une application de carnet d'adresses simple en utilisant Tkinter et les classes en Python. L'application doit permettre à l'utilisateur d'ajouter des contacts avec leurs noms et numéros de téléphone, et d'afficher la liste complète des contacts.

Définissez une classe Contact qui a des attributs pour le nom et le numéro de téléphone.

Créez une classe CarnetAdresses qui aura une liste de contacts et des méthodes pour ajouter un contact et récupérer la liste complète des contacts.

Utilisez Tkinter pour créer une interface utilisateur avec les éléments suivants :

Deux champs de saisie pour le nom et le numéro de téléphone.

Un bouton "Ajouter" pour ajouter le contact à la liste.

Une zone de texte pour afficher la liste des contacts.

Lorsque l'utilisateur clique sur le bouton "Ajouter", créez un objet Contact avec les informations saisies, ajoutez-le à la liste du carnet d'adresses, et mettez à jour la zone de texte avec la liste mise à jour.

Affichez le nom complet et le numéro de téléphone de chaque contact dans la zone de texte, chaque contact sur une nouvelle ligne.

Correction:

```
import tkinter as tk

# Étape 1: Définir la classe Contact
class Contact:
    def __init__(self, nom, numero):
        self.nom = nom
        self.numero = numero

    def __str__(self):
        return f"{self.nom} - {self.numero}"

# Étape 2: Créer la classe CarnetAdresses
class CarnetAdresses:
    def __init__(self):
```

```
        self.contacts = []

    def ajouter_contact(self, contact):
        self.contacts.append(contact)

    def get_liste_contacts(self):
        return self.contacts

# Étape 3: Construire l'interface utilisateur avec Tkinter
class Application(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Carnet d'Adresses")

        self.carnet = CarnetAdresses()

        tk.Label(self, text="Nom:").pack()
        self.nom_entry = tk.Entry(self)
        self.nom_entry.pack()

        tk.Label(self, text="Numéro:").pack()
        self.numero_entry = tk.Entry(self)
        self.numero_entry.pack()

        self.ajouter_btn = tk.Button(self, text="Ajouter", com-
mand=self.ajouter_contact)
        self.ajouter_btn.pack()

        self.contacts_text = tk.Text(self, height=10, width=50)
        self.contacts_text.pack()

    def ajouter_contact(self):
        nom = self.nom_entry.get()
        numero = self.numero_entry.get()
        if nom and numero:  # Vérifier que les champs ne sont pas vides
            contact = Contact(nom, numero)
            self.carnet.ajouter_contact(contact)
            self.mettre_a_jour_affichage()
            self.nom_entry.delete(0, tk.END)
            self.numero_entry.delete(0, tk.END)

    def mettre_a_jour_affichage(self):
        self.contacts_text.delete(1.0, tk.END)
        for contact in self.carnet.get_liste_contacts():
            self.contacts_text.insert(tk.END, str(contact) + "\n")

# Fonction d'exécution
def main():
    app = Application()
    app.mainloop()
```

```
if __name__ == "__main__":
    main()
```

Bon travail