Rapport des travaux pratiques

الكلية المتعددة التخصصات - بني ملال
Faculté Polydisciplinaire - Beni Mellal

PRÉSENTÉ À

PR.MOUNCIF

PRÉSENTÉ PAR

Younes Mouzait

Introduction :

Dans le cadre de ce rapport de travaux pratiques, nous avons exploré les concepts fondamentaux du langage Python à travers une série d'exercices pratiques. Python, reconnu pour sa simplicité et sa puissance, est un langage de programmation largement utilisé dans divers domaines, allant du développement web à l'intelligence artificielle.

L'objectif de ces travaux pratiques est de nous familiariser avec les structures de base du langage, telles que les listes, les boucles, les conditions et les fonctions. Ces exercices nous permettent de comprendre les principes de programmation.

Les concepts clés à travers ces exercices :

- Les types primitifs
- Les conditions
- Les boucles
- Les listes
- Les tuples
- Les dictionnaires
- La OOP
- Manipulation des fichiers
- Utilisation des bibliothèques

✓ Numpy

Travail Pratique 1

Exercice 1:

Écrire un programme, qui définit 3 variables : une variable de type texte, une variable de type nombre Entier, une variable de type nombre décimal et qui affiche leur type à l'aide de la fonction type(var).

```
Vra_text = input("donner une variable de type text ")
Vra_nombre = input("donner un nombre ")
Vra_décimal = input("donner une nombre décimal ")

print(Vra_text," est de type ",type(Vra_text))
print(Vra_nombre," est de type ",type(Vra_nombre))
print(Vra_décimal," est de type ",type(Vra_décimal))
```

Ecrire un exemple similaire pour affecter tes informations aux variables nom, prénom, Age .

```
nom,prenom,age = ('mouzait','younes',33)
```

Exercice 2:

A partir de la saisie des notes des matières (Anglais, Physique, Maths, Svt), et de leurs coefficients, calcule et affiche la moyenne des notes.

```
note_En = float(input("donner la note obtenue en Anglais"))
Coif_En = int(input("quel est le coefficient du Anglais"))
note_Ph = float(input("donner la note obtenue en Physique"))
Coif_Ph = int(input("quel est le coefficient du Physique"))
note_Ma = float(input("donner la note obtenue en Maths"))
Coif_Ma = int(input("quel est le coefficient du Maths"))
note_Svt = float(input("donner la note obtenue en SVT"))
Coif_Svt = int(input("quel est le coefficient du SVT"))

sum_coif = Coif_En + Coif_Ph + Coif_Ma + Coif_Svt
moyenne = (note_En*Coif_En + note_Ma*Coif_Ma + note_Ph*Coif_Ph +
note_Svt*Coif_Svt)/sum_coif
```

```
print("la mayenne des notes est : ",moyenne)
```

A partir de la saisie du budget une variable achats qui contient le montant de votre choix. Afficher si le budget permet de payer les achats.

```
achat = ("donner le montant de vos achats")
budget = ("donner le budget que vous avez")

if budget >= achat :
    print("vous pouvez payer vos achats")
else :
    print("vous ne pouvez pas payer vos achats")
```

méthode 2

```
message = "vous pouvez payer" if budget >= achat else "vous ne pouvez pas"
print(message)
```

Exercice 3

Écrire un programme qui, à partir de la saisie d'un rayon et d'une hauteur, calcule le volume d'un cône droit :

$$V = 1/3\ \pi r 2h$$

```
import math
rayon = float(input("donner le rayon du cone"))
hauteur = float(input("donner la hauteur du cone"))

volume = (1/3) * math.pi * rayon **2 * hauteur
print(f" le volume du cone est : {volume:.2f}")
```

Exercice 4

Une machine découpe dans une plaque, des disques circulaires de rayon Rg, percés d'un trou circulaire de rayon Rp avec Rp < Rg et ne débordant pas du disque. Quelle est la surface d'un disque découpé ?

```
import math
Rg = float(input("donner le rayon du grand disque"))
Rp = float(input("donner le rayon du petit disque"))
# calucle de la surface du grand disque
```

```
Sg = math.pi * Rg**2
# calucle de la surface du petit disque
Sp = math.pi * Rp**2
# la surface du disque découpé
S_trou = Sg - Sp
print(f"la surface du disque découpé est : {S_trou:.2f}")
```

Exercice 5

Ecrire un programme qui demande à l'utilisateur de saisir une phrase. Ensuite, le programme doit vérifier si la longueur de la phrase est paire ou impaire. Si la longueur est paire, le programme doit afficher la première moitié de la phrase. Si la longueur est impaire, le programme doit afficher la seconde moitié de la phrase.

```
phrase = input("donner un phrase")
l = len(phrase)
if l % 2 == 0 :
   print(phrase[:int(l/2)])
else :
   print(phrase[int(l/2):])
```

pour prendre la partie entière d’un nombre flottant on peut utiliser plusieurs méthodes :

```
int(l)
math.floor(l)
l = l//2
```

Travail Pratique 2

Exercice 1:

Écrire un code, qui définit une liste nommée Semaine qui contient les jours(lundi,..., dimanche) de la semaine, afficher la liste

```
semaine =["Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche"]
print(semaine)
```

Ecrire un code qui remplit la liste Couleurs par 5 couleurs différentes et affiche la liste

```
Couleur =[]
print("donner 5 couleurs différents")
for i in range(5) :
    Couleur.append(input())
print(Couleur)
```

Ecrire un code qui définit une liste de 7 réels et qui affiche les éléments ayant les indices 1, 3 et 5

```
Reels = [3.2,4.3,43.5,4.4,3,9.83,3.02]
print("Élément à l'indice 1 :",Reels[1])
print("Élément à l'indice 3 :",Reels[3])
print("Élément à l'indice 5 :",Reels[5])
```

Exercice 2

1- mylist = ['apple', 'banana', 'cherry']
print(mylist[1]) # affiche banana
2- mylist = ['apple', 'banana', 'banana', 'cherry']
print(mylist[2]) # affiche banana
3- thislist = ['apple', 'banana', 'cherry']
print(……..(thislist)) # print(len(thislist))
4- mylist = ['apple', 'banana', 'cherry']
print(mylist[-1]) # afficher le dernier élément cherry
5- fruits = ["apple", "banana", "cherry"]
print(fruits[1]) #Affiche le second élément de la liste
6- mylist = ['apple', 'banana', 'cherry', 'orange', 'kiwi']
print(mylist[1:4]) # banana , cherry , orange

7- fruits = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(fruits[3 : 6]) #Afficher le 3, 4 et 5 éléments de la liste

Exercice 3

Créer une liste de matières (Anglais, Physique, Maths, Svt) et affiche la liste.
Ajouter 2 nouvelles matières "Histoire" et "Geographie" à la liste puis afficher la liste
En utilisant l'opérateur [ ], afficher :
a. Les 4 premiers éléments de la liste .
b. Les 3 derniers éléments de la liste .
c. 3 éléments depuis le second indice .

```
Matiéres = ["Anglais","Physique","Maths","Svt"]
Matiéres.append("Histoire")
Matiéres.append("Geographie")
print(Matiéres)
print(Matiéres[:4])
print(Matiéres[3:])
print(Matiéres[1:4])
```

Exercice 4

1. mylist = ['apple', 'banana', 'cherry']
mylist[0] = 'kiwi'
print(mylist[1]) affichera banana
2. fruits = ["apple", "banana", "cherry"]
Modifiez la valeur de « apple » à « kiwi », dans la liste des fruits.

```
fruits[0] = "kiwi"
```

3. mylist = ['apple', 'banana', 'cherry']
mylist[1:2] = ['kiwi', 'mango']
print(mylist[2]) affichera mango
4. mylist = ['apple', 'banana', 'cherry']
mylist.insert(0, 'orange')
print(mylist[1]) affichera apple
5. fruits = ["apple", "banana", "cherry"]

#ajouter via la fonction « append » la valeur 'orange'

```
mylist = ['apple', 'banana', 'cherry']
mylist.append("orange")
```

6. fruits = ["apple", "banana", "cherry"] #utiliser « insert »pour ajouter à la seconde position « lemon »

```
mylist = ['apple', 'banana', 'cherry']
mylist.insert(1,"orange")
print(mylist)
```

Exercice 5

utilise la méthode « remove » pour supprimer la valeur « mer » de la liste Semaine=['lun', 'mar', 'mer', 'jeu', 'ven', 'sam', 'dim']

```
Semaine=['lun', 'mar', 'mer', 'jeu', 'ven', 'sam', 'dim']
Semaine.remove('mer')
print(Semaine)
```

2. mylist = ['apple', 'banana', 'cherry'] mylist.pop(1) print(mylist)

Resultat :

```
['apple', 'cherry']
```

3. fruits = ['apple', 'banana', 'cherry']

Quelle est la fonction qui permet de vider la liste del, pop, clear , delete

```
fruits.clear()
```

Travail Pratique 3

Exercice 1

Écrire une liste des nombres suivants : 4, 8, 15, 16, 23, 42

Ecrire une fonction affListe qui prend une liste en paramètre et affiche chaque élément de la liste.

```
nombres = [4,8,15,16,23,42]
# premiére méthode
def affListe(list) :
      for i in list :
          print(i,end =" ")
affListe(nombres)
```

Exercice 2

Ecris une fonction somme_liste(liste) qui retourne la somme des éléments d'une liste

Ajoute une autre fonction moyenne_liste(liste) qui calcule et retourne la moyenne des éléments

Utilise ces fonctions pour calculer la somme et la moyenne des éléments de la liste nombres

```
nombres = [4,8,15,16,23,42]
# calcule la somme
def somme(list) :
    s = 0
    for i in list :
        s=i
    return i
# calcule la moyenne
def moyenne(list) :
    return somme(list)/len(list)

print(" la somme est : ",somme(nombres)," la moyenne est :
",moyenne(nombres))
```

Exercice 3

1- Écris une fonction extraire_pairs(liste) qui retourne une nouvelle liste contenant uniquement les nombres pairs de la liste donnée.

2- Teste cette fonction avec la liste nombres.

3- Affiche la liste des nombres pairs retournée.

```
nombres = [4,8,15,16,23,42]

def extraire_pairs(list) :
    result =[]
    for i in list :
        if i%2 == 0 :
            result.append(i)
    return result

print(extraire_pairs(nombres))
```

Exercice 4

Écris une fonction element_existe(liste, element) qui vérifie si un élément donné existe dans une liste. La fonction doit retourner True si l'élément est présent, sinon False.

Teste cette fonction avec :

- L'élément 15 dans la liste nombres.
- L'élément 50 dans la liste nombres.

```
nombres = [4,8,15,16,23,42]

def element_exist(list,element):
    return element in list

print(element_exist(nombres,15))
print(element_exist(nombres,50))
```

Exercice 5

Écris une fonction liste_carres(liste) qui retourne une nouvelle liste contenant les carrés des éléments de la liste d'entrée.

Teste cette fonction avec la liste nombres.

Affiche la liste des carrés retournée.

```
nombres = [4,8,15,16,23,42]

def list_carres(list) :
    result =[]
    for i in list :
        result.append(i**2)
    return result
print(list_carres(nombres))
```

Exercice 6

Écris une fonction trouver_min_max(liste) qui retourne le minimum et le maximum d'une liste sous forme d'un tuple (min, max)

Teste cette fonction avec la liste nombres.

```python
nombres = [4,8,15,16,23,42]

def trouver_min_max(liste):
    return max(liste),min(liste)

print(trouver_min_max(nombres))
```

Exercice 7

Crée une deuxième liste appelée autres_nombres contenant : 7, 11, 19, 24, 33.
Écris une fonction fusionner_et_trier(liste1, liste2) qui fusionne deux listes et retourne une nouvelle liste triée.
Teste cette fonction avec nombres et autres_nombre .

```python
nombres = [4,8,15,16,23,42]
autres_nombres =[7,11,19,24,33]

def fusionner_et_trier(list1,list2):
    return list1+list2

print(fusionner_et_trier(nombres,autres_nombres))
```

Exercice 8

Écris une fonction est_palindrome(mot) qui vérifie si un mot donné est un palindrome (il se lit de la même façon dans les deux sens).
Teste cette fonction avec les mots suivants :

"radar" "python" "level"

```python
def est_palindrome(list):
    list1 = list[::-1]
    return list == list1

print(est_palindrome("radar"))
print(est_palindrome("python"))
print(est_palindrome("level"))
```

Travail Pratique 4

Exercice 1 : Manipulation de dictionnaires

Créez un programme qui :
1. Demande à l'utilisateur d'entrer des noms de personnes et leur âge, sous la forme `Nom:Âge`.
2. Stocke ces informations dans un dictionnaire.
3. Permet à l'utilisateur de rechercher un nom et d'afficher son âge.
4. Ajoute une fonctionnalité pour supprimer une personne du dictionnaire.

```
dictionnaire ={}
def ajouter():
    Nom = input("entrer le nom de la personne : " )
    Age = int(input("entrer l'age de la personne : "))
    dictionnaire[Nom]=Age

def chercher():
    Nom = input("entrer le nom que vous cherchéez " )
    if Nom in dictionnaire:
        print(dictionnaire[Nom])
    else:
        print("le nom n'existe pas")

def supprimer():
    nom= input("entrer le nom à supprimer : ")
    if nom in dictionnaire:
        del dictionnaire[nom]
    else:
        print(f"{nom} n'existe pas dans le dictionnaire")


fois  =int(input("entrer le nombre de valeurs que vous voulez saisir : "))
for i in range(fois):
    ajouter()

chercher()
supprimer()
print(dictionnaire)
```

Exercice 2 : Fusion et tri de dictionnaires

1. Écrivez une fonction qui prend deux dictionnaires et retourne un seul dictionnaire fusionné.
2. Les clés doivent être triées par ordre alphabétique.
3. Si deux clés identiques existent, leurs valeurs doivent être additionnées (si elles sont numériques) ou concaténées (si elles sont des chaînes).

```
def fusionner_et_trier(D1, D2):
    fusion = {}
    for d in (D1, D2):
        for key, value in d.items():
            if key in fusion:
                fusion[key] = fusion[key] + value if isinstance(value,
(int, float)) else str(fusion[key]) + str(value)
            else:
                fusion[key] = value
    return dict(sorted(fusion.items()))

d1 = {"a": 10, "b": "test"}
d2 = {"a": 5, "c": "data"}

print(fusionner_et_trier(d1, d2))
```

Exercice 3 : Lecture et écriture dans un fichier

1. Créez un fichier texte contenant une liste de noms et d'âges (un par ligne, format `Nom,Âge`).
2. Écrivez un programme qui lit ce fichier, stocke les données dans un dictionnaire, puis affiche ce dictionnaire.
3. Ajoutez une option pour modifier ou ajouter un enregistrement dans le fichier.

```
with open("noms_et_ages.txt", "w") as f:
    f.write("Ahmed,25\nAdil,30\nYassine,22\n")
# Lecture et stockage dans un dictionnaire
def lire_fichier():
    dictionnaire = {}
    with open("noms_et_ages.txt", "r") as f:
        for ligne in f:
            nom, age = ligne.strip().split(",")
```

```
            dictionnaire[nom] = int(age)
    return dictionnaire
print(lire_fichier())
```

Exercice 4 : Analyse de données depuis un fichier

1. Donnez un fichier contenant des notes d'étudiants au format `Nom,Note`.
2. Créez un programme qui :
- Lit les données du fichier.
- Calcule la note moyenne.
- Affiche les noms des étudiants ayant une note supérieure à la moyenne.

```
# Création du fichier
with open("notes.txt", "w") as f:
    f.write("Hassan,15\nAhmed,12\nKhalid,17\n")
# Calcul de la moyenne
def calculer_moyenne():
    notes = []
    with open("notes.txt", "r") as f:
        for ligne in f:
            note = ligne.strip().split(",")
            notes.append(int(note))
    moyenne = sum(notes) / len(notes)
    for ligne in f:
        if note > moyenne :
            print(nom)
    return moyenne
```

Exercice 5 : Introduction à la POO - Classe Étudiant

1. Définissez une classe `Etudiant` avec les attributs `nom`, `âge` et `note`.
2. Implémentez une méthode `afficher_info()` qui affiche le informations de l'étudiant.
3. Créez une méthode statique pour calculer la moyenne des notes d'une liste d'objets `Etudiant`.

```
class Etudiant:
    def __init__(self, nom, age, note):
        self.nom = nom
        self.age = age
        self.note = note
```

```
    def afficher_info(self):
        print(f"Nom : {self.nom}, Âge : {self.age}, Note : {self.note}")

    @staticmethod
    def moyenne_notes(etudiants):
        return sum(e.note for e in etudiants) / len(etudiants)
etudiants = [Etudiant("hiba", 20, 15), Etudiant("kamal", 18, 16)]
Etudiant1= Etudiant("hiba", 20, 15)
Etudiant2= Etudiant("kamal", 18, 16)
print(f"Moyenne des notes est : {Etudiant.moyenne_notes(etudiants)}")
print(f" les étudiants inscrits sont : \n")
print(Etudiant1.afficher_info())
print(Etudiant2.afficher_info())
```

Exercice 6 : Gestion d'un carnet d'adresses

1. Créez une classe `CarnetAdresses` qui utilise un dictionnaire pour stocker des informations
(nom, email, téléphone).
2. Ajoutez des méthodes pour :
- Ajouter un contact.
- Supprimer un contact.
- Rechercher un contact par nom.
3. Sauvegardez les contacts dans un fichier pour les recharger à la prochaine exécution du programme.

```
import csv
class CarnetAdresses:
    def __init__(self):
        self.contacts = {}
    def ajouter_contact(self, nom, email, telephone):
        self.contacts[nom] = {"email": email, "telephone": telephone}
    def supprimer_contact(self, nom):
        self.contacts.pop(nom, None)
    def rechercher_contact(self, nom):
        return self.contacts.get(nom, "Contact non trouvé")
    def afficher_contacts(self):
        return self.contacts
    def sauvegarder_contacts(self, fichier):
        with open(fichier, "w", encoding="utf-8") as f:
            for nom, (email, telephone) in self.contacts.items():
                f.write(f"{nom},{email},{telephone}\n")
```

```
carnet = CarnetAdresses()
carnet.ajouter_contact("bibal", "bibal@gmail.com", "0625235325")
carnet.ajouter_contact("ahmed", "ahmed@gmail.com", "0621237325")
print(carnet.rechercher_contact("bibal"))
print("\n", carnet.afficher_contacts())
carnet.supprimer_contact("bibal")
print("\n", carnet.afficher_contacts())
carnet.sauvegarder_contacts("contacts.txt")
```

Exercice 7 : Simulation d'une bibliothèque

1. Créez une classe `Livre` avec les attributs `titre`, `auteur` et `statut` (disponible/emprunté).
2. Implémentez une classe `Bibliotheque` qui contient une liste de livres.
3. Ajoutez des méthodes pour :
- Ajouter un livre.
- Emprunter un livre (changer le statut à "emprunté").
- Rendre un livre (changer le statut à "disponible").
- Lister les livres disponibles.

```
class Livre:
    def __init__(self, titre, auteur, statut="disponible"):
        self.titre = titre
        self.auteur = auteur
        self.statut = statut

    def emprunter(self):
        if self.statut == "disponible":
            self.statut = "emprunté"

    def rendre(self):
        if self.statut == "emprunté":
            self.statut = "disponible"

class Bibliotheque:
    def __init__(self):
        self.livres = []

    def ajouter_livre(self, livre):
        self.livres.append(livre)

    def lister_disponibles(self):
```

```
        return [livre.titre for livre in self.livres if livre.statut ==
"disponible"]

biblio = Bibliotheque()
biblio.ajouter_livre(Livre("Python", "Auteur1"))
print(biblio.lister_disponibles())
```

Travail Pratique 5

Exercice 1 : Création et Manipulation de Tableaux

1. Créez un tableau 1D contenant les nombres de 0 à 9.
- Créez un tableau 2D de dimensions (3, 3) avec des nombres aléatoires entre 0 et 1.
- Créez un tableau 3D de dimensions (2, 3, 4) rempli de zéros.
2. Accédez au troisième élément du tableau 1D.
- Accédez à la deuxième ligne et première colonne du tableau 2D.
- Modifiez un élément du tableau 3D.
3. Récupérez les trois premiers éléments du tableau 1D et prenez la dernière colonne du tableau 2D.

```
import numpy as np

# 1. Création des tableaux
# Tableau 1D de 0 à 9
tab1d = np.arange(10)

# Tableau 2D (3x3) avec valeurs aléatoires entre 0 et 1
tab2d = np.random.rand(3, 3)

# Tableau 3D (2x3x4) rempli de zéros
tab3d = np.zeros((2, 3, 4))

print("1. Tableaux créés :")
print("1D :", tab1d)
print("2D :\n", tab2d)
print("3D :\n", tab3d)

# 2. Accès aux éléments
# Troisième élément du 1D (index 2)
```

```python
elem1d = tab1d[2]

# Deuxième ligne, première colonne du 2D
elem2d = tab2d[1, 0]

# Modification d'un élément du 3D (premier plan, 2e ligne, 3e colonne)
tab3d[0, 1, 2] = 5.0

print("\n2. Accès et modification :")
print("3e élément 1D :", elem1d)
print("2e ligne, 1e colonne 2D :", elem2d)
print("3D modifié :\n", tab3d)

# 3. Extraction de données
# Trois premiers éléments du 1D
premiers = tab1d[:3]

# Dernière colonne du 2D
derniere_col = tab2d[:, -1]

print("\n3. Extraction :")
print("3 premiers 1D :", premiers)
print("Dernière colonne 2D :", derniere_col)
# Modification d'un élément du 3D (premier plan, 2e ligne, 3e colonne)
tab3d[0, 1, 2] = 5.0

print("\n2. Accès et modification :")
print("3e élément 1D :", elem1d)
print("2e ligne, 1e colonne 2D :", elem2d)
print("3D modifié :\n", tab3d)

# 3. Extraction de données
# Trois premiers éléments du 1D
premiers = tab1d[:3]

# Dernière colonne du 2D
derniere_col = tab2d[:, -1]

print("\n3. Extraction :")
print("3 premiers 1D :", premiers)
print("Dernière colonne 2D :", derniere_col)
```

Exercice 2 : Opérations Mathématiques et Logiques

1. Créez deux tableaux 1D de longueur 5 et effectuez l'addition, la soustraction, la
multiplication et la division élément par élément.
2. Appliquez `np.sin`, `np.cos`, et `np.exp` sur un tableau de valeurs entre 0 et π.
2. Créez un tableau d'entiers de longueur 10.
- Sélectionnez tous les éléments pairs.
- Remplacez tous les éléments impairs par -1.

```
import numpy as np
# 1. Opérations élémentaires sur deux tableaux 1D
print("=== Partie 1 ===")
a = np.array([1, 2, 3, 4, 5])
b = np.array([5, 4, 3, 2, 1])

print("Addition :", a + b)
print("Soustraction :", a - b)
print("Multiplication :", a * b)
print("Division :", a / b)  # Attention aux divisions par zéro si b
contient 0
# 2. Fonctions mathématiques sur un tableau de 0 à π
print("\n=== Partie 2 ===")
angles = np.linspace(0, np.pi, 5)  # 5 valeurs entre 0 et π
print("Valeurs originales :", angles)
print("Sinus :", np.sin(angles))
print("Cosinus :", np.cos(angles))
print("Exponentielle :", np.exp(angles))

# 3. Manipulation d'un tableau d'entiers
print("\n=== Partie 3 ===")
arr = np.arange(10)  # Tableau de 0 à 9
print("Tableau original :", arr)

# Sélection des éléments pairs
pairs = arr[arr % 2 == 0]
print("Éléments pairs :", pairs)

# Remplacement des impairs par -1
arr_modifie = arr.copy()
arr_modifie[arr % 2 != 0] = -1
print("Tableau modifié :", arr_modifie)
```

Exercice 3 : Reshaping et Axes

1. Créez un tableau de 12 éléments.
- Transformez-le en un tableau 2D de dimensions (3, 4).
- Puis, en un tableau 3D de dimensions (2, 2, 3).
2. Transposez le tableau 2D obtenu précédemment et appliquez `np.swapaxes` et observez les résultats.
3. Créez deux tableaux 2D de dimensions (2, 3).
- Concaténez-les verticalement et horizontalement.
- Divisez le tableau concaténé en sous-tableaux .

```
import numpy as np

# 1. Création et transformations du tableau
print("=== Partie 1 ===")
# Création d'un tableau 1D de 12 éléments
tab1d = np.arange(12)
print("Tableau 1D original :\n", tab1d)

# Transformation en 2D (3x4)
tab2d = tab1d.reshape((3, 4))
print("\nTableau 2D (3x4) :\n", tab2d)

# Transformation en 3D (2x2x3)
tab3d = tab1d.reshape((2, 2, 3))
print("\nTableau 3D (2x2x3) :\n", tab3d)

# 2. Opérations sur le tableau 2D
print("\n=== Partie 2 ===")
# Transposition du tableau 2D
transpose = tab2d.T
print("Transposée (4x3) :\n", transpose)

# Échange des axes 0 et 1
swap = np.swapaxes(tab2d, 0, 1)
print("\nSwap axes (4x3) :\n", swap)

# 3. Concaténation et division
print("\n=== Partie 3 ===")
# Création de deux tableaux 2D
a = np.array([[1, 2, 3], [4, 5, 6]])
b = np.array([[7, 8, 9], [10, 11, 12]])
```

```python
# Concaténation verticale (axis=0)
vstack = np.vstack((a, b))
print("Concaténation verticale :\n", vstack)

# Concaténation horizontale (axis=1)
hstack = np.hstack((a, b))
print("\nConcaténation horizontale :\n", hstack)

# Division du tableau concaténé
split_v = np.vsplit(vstack, 2)  # Division verticale
print("\nDivision verticale :")
for i, arr in enumerate(split_v):
    print(f"Partie {i+1}:\n", arr)

split_h = np.hsplit(hstack, 3)  # Division horizontale
print("\nDivision horizontale :")
for i, arr in enumerate(split_h):
    print(f"Partie {i+1}:\n", arr)

import numpy as np
arr1=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
arr2=arr1.reshape(3,4)
arr3=arr1.reshape(2,2,3)
# 2
A3 = arr2.T
arr4=np.swapaxes(arr2,0,1)
print(arr1)
print(arr2)
print(arr3)
print("Transposé", A3)
print(arr4)
```

Exercice 4 : Fonctions Avancées

1. Créez un tableau aléatoire de dimensions (5, 5), puis calculez la moyenne, l'écart-type, le
minimum et le maximum par ligne et par colonne.
2. Créez un tableau de nombres aléatoires, triez-le dans l'ordre croissant et trouvez l'indice de l'élément maximum.

3. Créez un tableau 1D de longueur 4 et un tableau 2D de dimensions (3, 4) puis multipliez-les en utilisant le broadcasting.

```
import numpy as np
#Q1
arr1 = np.random.rand(5,5)
moyenne1 = np.mean(arr1,axis=0)
ecart_type1 = np.std(arr1,axis=0)
max1 = np.max(arr1,axis=0)
min1 = np.min(arr1,axis=0)
moyenne2 = np.mean(arr1,axis=1)
ecart_type2 = np.std(arr1,axis=1)
max2 = np.max(arr1,axis=1)
min2 = np.min(arr1,axis=1)
#Q2
tableau_aleatoire = np.random.rand(10)
ord_croi=np.sort(tableau_aleatoire)
max_element=np.argmax(tableau_aleatoire)
#Q3
tableau_1D = np.array([1, 2, 3, 4])
tableau_2D = np.array([[1, 2, 3, 4],[5, 6, 7, 8],[9, 10, 11, 12]])
resultat = tableau_2D * tableau_1D
print("Tableau aléatoire :")
print(arr1)
print("Moyenne par ligne :", moyenne1)
print("Écart-type par ligne :", ecart_type1)
print("Minimum par ligne :", min1)
print("Maximum par ligne :", max1)
print("Moyenne par colonne :", moyenne2)
print("Écart-type par colonne :", ecart_type2)
print("Minimum par colonne :", min2)
print("Maximum par colonne :", max2)
print( resultat)
```

Tableau aléatoire :
[[0.87292895 0.88216377 0.52677899 0.54701782 0.28140767]
 [0.28233356 0.23672636 0.25936342 0.66093459 0.74204776]
 [0.98715956 0.49346944 0.61976733 0.6271538 0.48232146]
 [0.13930055 0.79012546 0.29063581 0.60247055 0.14897176]
 [0.63001724 0.88074335 0.58276621 0.37701168 0.317459]]
Moyenne par ligne : [0.58234797 0.65664568 0.45586235 0.56291769 0.39444153]
Écart-type par ligne : [0.32768331 0.25371391 0.15093723 0.10011181 0.20372106]

Minimum par ligne : [0.13930055 0.23672636 0.25936342 0.37701168 0.14897176]
Maximum par ligne : [0.98715956 0.88216377 0.61976733 0.66093459 0.74204776]
Moyenne par colonne : [0.62205944 0.43628114 0.64197432 0.39430082 0.5575995]
Écart-type par colonne : [0.22862328 0.21853341 0.18297999 0.25921794 0.20028932]
Minimum par colonne : [0.28140767 0.23672636 0.48232146 0.13930055 0.317459]
Maximum par colonne : [0.88216377 0.74204776 0.98715956 0.79012546 0.88074335]
[[1 4 9 16]P
[5 12 21 32]
[9 20 33 48]]

Exercice 5 :

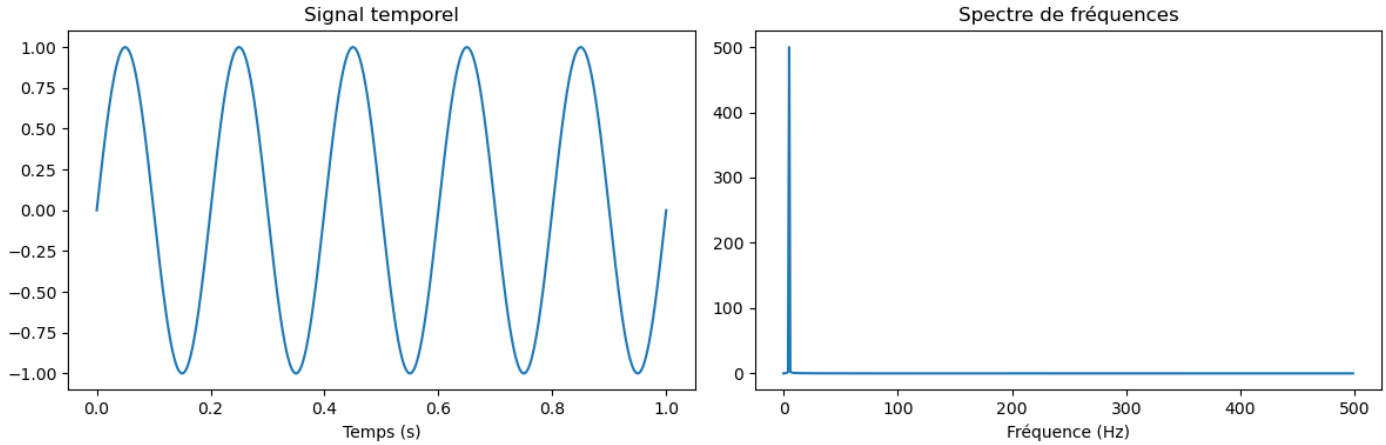
1. Créez un tableau de données de dimensions (100, 3) où chaque colonne représente une variable aléatoire et calculez la matrice de covariance.
2. Appliquez la transformation de Fourier sur un tableau de données sinusoïdales et affichez le spectre de fréquences.
3. Simulez 1000 lancers de deux dés et affichez l'histogramme des sommes obtenues.

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

# 1. Matrice de covariance
print("=== Partie 1 : Matrice de covariance ===")
data = np.random.randn(100, 3)  # 100 observations, 3 variables
cov_matrix = np.cov(data, rowvar=False)  # rowvar=False car les variables sont en colonnes

print("Matrice de covariance (3x3) :\n", cov_matrix)

# 2. Transformée de Fourier
print("\n=== Partie 2 : Transformée de Fourier ===")
freq = 5  # Fréquence du signal (Hz)
t = np.linspace(0, 1, 1000)  # 1 seconde d'échantillonnage
signal = np.sin(2 * np.pi * freq * t)  # Signal sinusoïdal

fft_result = np.fft.fft(signal)  # Transformée de Fourier
frequencies = np.fft.fftfreq(len(t), t[1] - t[0])  # Axe des fréquences

plt.figure(figsize=(12, 4))
plt.subplot(121)
plt.plot(t, signal)
plt.title("Signal temporel")
plt.xlabel("Temps (s)")
```

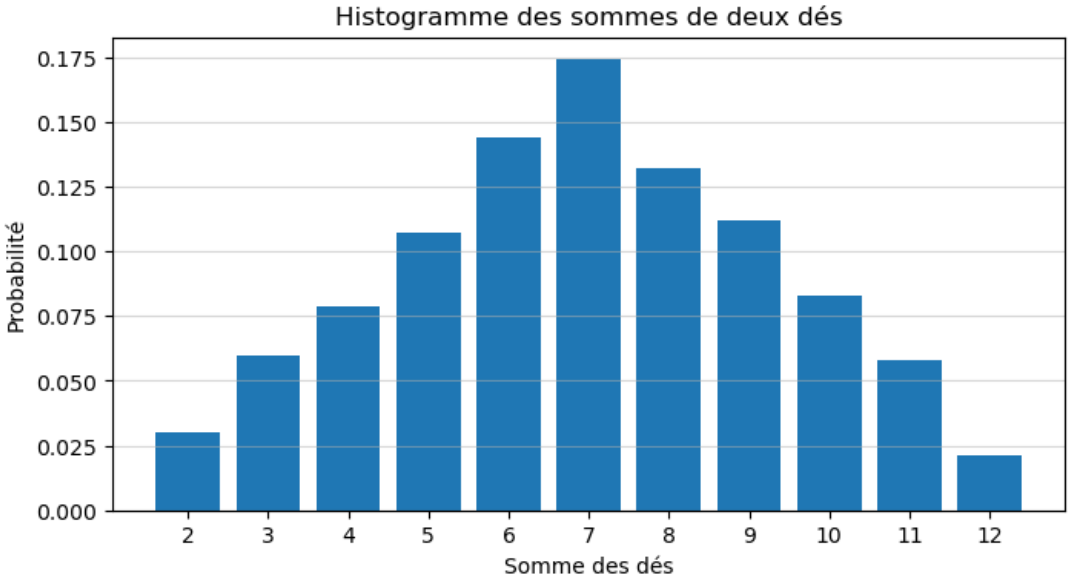
```
plt.subplot(122)
plt.plot(frequencies[:len(frequencies)//2],
np.abs(fft_result)[:len(fft_result)//2])  # Spectre unilateral
plt.title("Spectre de fréquences")
plt.xlabel("Fréquence (Hz)")
plt.tight_layout()
plt.show()

# 3. Simulation de lancers de dés
print("\n=== Partie 3 : Lancers de dés ===")
np.random.seed(42)  # Pour la reproductibilité
lancers_de1 = np.random.randint(1, 7, 1000)
lancers_de2 = np.random.randint(1, 7, 1000)
sommes = lancers_de1 + lancers_de2

plt.figure(figsize=(8, 4))
plt.hist(sommes, bins=range(2, 14), align='left', rwidth=0.8,
density=True)
plt.title("Histogramme des sommes de deux dés")
plt.xlabel("Somme des dés")
plt.ylabel("Probabilité")
plt.xticks(range(2, 13))
plt.grid(axis='y', alpha=0.5)
plt.show()
```

=== Partie 1 : Matrice de covariance ===
Matrice de covariance (3x3) :
[[1.18752505 -0.00771493 0.10030757]
 [-0.00771493 1.16584658 0.03848563]
 [0.10030757 0.03848563 1.01137553]]

=== Partie 2 : Transformée de Fourier ===



=== Partie 3 : Lancers de dés ===



Exercice 6

Considérons un ensemble de données représentant les ventes mensuelles de trois produits (P1, P2 et P3) au cours d'une année.
1. Créer un tableau NumPy 2D avec 12 lignes (pour chaque mois) et 3 colonnes (pour les produits P1, P2 et P3). Utilisez des valeurs aléatoires entre 100 et 1000 pour représenter les ventes.
2. Calculer et afficher le total des ventes pour chaque produit sur l'ensemble de l'année.
3. Calculer et afficher la moyenne des ventes mensuelles pour chaque produit.
4. Identifier le mois ayant les ventes maximales pour chaque produit.
5. Calculer la croissance mensuelle en pourcentage par rapport au mois précédent pour chaque produit. Taux de croissance = ((Vente mois N – Vente mois N-1) / Vente mois N-1) x 100 Utiliser la fonction np.diff()
1. Identifier et afficher le mois avec la plus forte croissance pour chaque produit.
2. Créer un tableau 1D contenant la somme des ventes pour chaque mois (total des ventes tous produits confondus).

```
import numpy as np
#Q1
tab= np.random.randint(100,1000,size=(12,3))
#Q2
totaux_produits = np.sum(tab, axis=0)
```

```
print(f"Total des ventes : P1 = {totaux_produits[0]}, P2 =
{totaux_produits[1]}, P3 = {totaux_produits[2]}")
#Q3
moyenne_produits = np.mean(tab, axis=0)
print(f"Moyenne des ventes : P1 = {moyenne_produits[0]:.2f}, P2 =
{moyenne_produits[1]:.2f}, P3 = {moyenne_produits[2]:.2f}")
#Q4
mois_max_ventes = np.argmax(tab, axis=0)
print(f"Mois avec ventes maximales : P1 = Mois {mois_max_ventes[0] + 1},
P2 = Mois {mois_max_ventes[1] + 1}, P3 = Mois {mois_max_ventes[2] + 1}")
Total des ventes : P1 = 6667, P2 = 5852, P3 = 6842
Moyenne des ventes : P1 = 555.58, P2 = 487.67, P3 = 570.17
Mois avec ventes maximales : P1 = Mois 11, P2 = Mois 8, P3 = Mois 10
```