



الكلية المتعددة التخصصات - بني ملال
Faculté Polydisciplinaire - Beni Mellal



Compte rendu :

TP Python :

1,2,3,4 et 5



Etudiante : SARA ELHAMRI
Enseignant : Mr.Prof.MONCIF
Filière : IAI3D

Introduction :

Python est un langage de programmation simple et puissant, utilisé pour créer des programmes dans des domaines comme le développement web, l'analyse de données, l'automatisation et les jeux. Sa syntaxe claire le rend facile à apprendre, même pour les débutants. Il est très populaire en raison de sa flexibilité et de ses nombreuses bibliothèques utiles.

TP1 en Python:

Exercice 1:

1/Ecriture d'un programme qui définit 3 variables :

```
exer1tp1.py > ...  
1  '''déclaration des variables'''  
2  str1='Bonjour'  
3  int1=12  
4  dec1=13,5  
5
```

```
"""affichage le type des variables"""  
print("type de la variable str1 est:",type(str1))  
print(int1,"est de type ",type(int1))  
print(dec1,"est de type",type(dec1))
```

2/Ecriture d'un exemple similaire :

```
"""ecrire un exemple"""  
nom,prenom,age=("Ali","Adam",23)  
print("le nom est :",nom)  
print("le prenom est :",prenom)  
print("l'age est:",age)
```



```
PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
  
PS C:\Users\EL HAMRI\Documents\vscode exercice> ^C  
PS C:\Users\EL HAMRI\Documents\vscode exercice> & "C:/Users/EL HAMRI/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe" C:\Users\EL HAMRI\Documents\vscode exercice/exer1tp1.py  
type de la variable str1 est: <class 'str'>  
12 est de type <class 'int'>  
(13, 5) est de type <class 'tuple'>  
le nom est : Ali  
le prenom est : Adam  
l'age est: 23  
PS C:\Users\EL HAMRI\Documents\vscode exercice> █
```

Exercice 2:

1/Calcul et affichage de la moyenne des notes :

```
exer2tp1.py > coef
1  """calcul et affiche la moyenne des notes"""
2  math=input("entrer math result")
3  physique=input("entrer physique result")
4  svt=input("entrer svt result")
5  anglais=input("entrer anglais result")
6  coef-math=input("entrer coef-math")
```

2/calcul du budget des achats :

```
coef-physique=input("entrer coef-physique")
coef-svt=input("entrer coef-svt")
coef-anglais=input("entrer coef-anglais")
total-coef=int(coef-math)+int(coef-physique)+int(coef-svt)+int(coef-anglais)
matier avec coef=float(math)*int(coef-math)+float(physique)*int(coef-physique)+float(svt)*int(coef-svt)+float(anglais)*int(coef-anglais)
moyenne=matiere avec coef/total coef
```



```
PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL PORTS
• PS C:\Users\EL HAMRI\Documents\vscode exercice> & "C:/Users/EL HAMRI\Documents\vscode exercice/exer4tp1.py"
  entrer le rayon du disque5
  entrer le rayon du trou3
  la surface du disque découpé est : 50.239999999999995
○ PS C:\Users\EL HAMRI\Documents\vscode exercice>
```

Exercise 3:

```
1 """demander à l'utilisateur de saisir le rayon et la hauteur"""
2 rayon=float(input("entrer le rayon: "))
3 hauteur = float(input("entrer la hauteur : "))
4
5 """calcule le volume"""
6 v=1/3*3.14*rayon**2*hauteur
7
8 """affichage du resultat"""
9 print("le volume est:"+str(v))
10
```



```

PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\EL HAMRI\Documents\vscode exercice> ^C
PS C:\Users\EL HAMRI\Documents\vscode exercice> & "C:/Users/EL HAMRI/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe" "C:/Users/EL HAMRI/AppData/Local/Programs/Python/Python39-64/Scripts/exer1tp1.py"
type de la variable str1 est: <class 'str'>
12 est de type <class 'int'>
(13, 5) est de type <class 'tuple'>
le nom est : Ali
le prenom est : Adam
l'age est: 23
PS C:\Users\EL HAMRI\Documents\vscode exercice> 
```

Exercise 4 :

```
exer4tp1.py / ...
1 rg=float(input("entrer le rayon du disque"))
2 rp=float(input("entrer le rayon du trou"))
3 if rg>rp:
4     S=3.14*rg**2-3.14*rp**2
5     print("la surface du disque découpé est : " , S)
6 else:
7     print("vous ne pouvez pas découper le disque")
```

Exercice 5 :

```
exerc5tp1.py > ...  
1 phrase=input("veuillez saisir la phrase")  
2 longueur=len(phrase)  
3 if longueur %2==0:  
4     print("la longueur de la phrase est paire")  
5 else:  
6     print("la longueur de la phrase est impaire")
```



PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\EL HAMRI\Documents\vscode exercice> ^C  
● PS C:\Users\EL HAMRI\Documents\vscode exercice> & "C:/Users/EL HAMRI/Applications/Python/Python38-64/Scripts/python.exe" c:\Users\EL HAMRI\Documents\vscode exercice/exer5tp1.py  
veuillez saisir la phrasele garçon mange la pomme  
la longueur de la phrase est paire  
○ PS C:\Users\EL HAMRI\Documents\vscode exercice> |
```

TP2 en Python:

Exercice 1:

1/Définition et affichage d'une liste :

```
exerc1tp2.py / ...  
1  semaine=["lundi","mardi","mercredi","jeudi","vendredi","samedi","dimanche"]  
2  print(semaine)  
3  for jour in semaine:  
4      print(jour)
```



```
PS C:\Users\EL HAMRI\Documents\vscode exercice> & "C:/Users/EL HAMRI/AppData/L  
cice/exerc1tp2.py"  
['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi', 'samedi', 'dimanche']  
lundi  
mardi  
mercredi  
jeudi  
vendredi  
samedi  
dimanche  
PS C:\Users\EL HAMRI\Documents\vscode exercice>
```

2/Affichage de la liste couleurs :

```
"""2/"""  
couleurs=["rouge","jaune","vert","bleu","noir"]  
"""affichage"""  
print(couleurs)  
"""affichage par boucle for"""  
for color in couleurs:  
    print(color)
```



```
['rouge', 'jaune', 'vert', 'bleu', 'noir']  
rouge  
jaune  
vert  
bleu  
noir
```

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

```
1 Nombres = [1.5, 2.3, 3.7, 4.1, 5.9, 6.8, 7.2]  
2 print(Nombres[1], Nombres[3], Nombres[5])  
→
```



```
2.3 4.1 6.8
```

Exercice 2:

```
# 1. Affichage du deuxième élément  
mylist = ['apple', 'banana', 'cherry']  
print(mylist[1]) # Résultat : banana  
# 2. Affichage du troisième élément  
mylist = ['apple', 'banana', 'banana', 'cherry']  
print(mylist[2]) # Résultat : banana  
  
# 3. Taille de la liste  
thislist = ['apple', 'banana', 'cherry']  
print(len(thislist))  
# 4. Dernier élément de la liste  
mylist = ['apple', 'banana', 'cherry']  
print(mylist[-1]) # Résultat : cherry  
# 5. Affichage du deuxième élément  
fruits = ["apple", "banana", "cherry"]  
print(fruits[1]) # Résultat : banana  
  
# 6. Extraction d'une sous-liste  
mylist = ['apple', 'banana', 'cherry', 'orange', 'kiwi']  
print(mylist[1:4]) # Résultat : ['banana', 'cherry', 'orange']  
# 7. Affichage du 3e, 4e et 5e élément  
fruits = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(fruits[2:5]) # Résultat : ['cherry', 'orange', 'kiwi']
```




```
cherry
banana
['banana', 'cherry', 'orange']
['cherry', 'orange', 'kiwi']
```



```
banana
banana
3
```

Exercice 3:

```
exe3tp2.py > ...
1  matieres=['anglais','physiques','maths','svt']
2  #affichage
3  print(matieres)
4  for m in matieres:
5      |    print(m)
6
7
8  #ajout de matieres
9  matieres.append('histoires')
10 #ajout en utilisant insert()
11 matieres.insert(2,'geographie')
12 print (matieres)
13
14
15 #utilisant operateur []
16 print(matieres[:4])
17 print(matieres[-3:])
18 print(matieres[2:5])
19 |
20
21
```



```
PS C:\Users\EL HAMRI\Documents\vscode-exercice> python C:\Users\EL HAMRI\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.10.0.0_q-bcs88f6h-6q93x-qeuevaq09z0j6lq?app=Python.exe
exercice/exe3tp2.py
['anglais', 'physiques', 'maths', 'svt']
anglais
physiques
maths
svt
['anglais', 'physiques', 'geographie', 'maths', 'svt', 'histoires']
['anglais', 'physiques', 'geographie', 'maths']
['maths', 'svt', 'histoires']
['geographie', 'maths', 'svt']
PS C:\Users\EL HAMRI\Documents\vscode-exercice>
```

Exercice 4:

```
# 1. Modification d'un élément
mylist = ['apple', 'banana', 'cherry']
mylist[0] = 'kiwi'
print(mylist[1]) # Résultat : banana
# 2. Remplacement de "apple" par "kiwi"
fruits = ["apple", "banana", "cherry"]
fruits[0] = "kiwi"
print(fruits)
# 3. Remplacement d'un élément par plusieurs
mylist = ['apple', 'banana', 'cherry']
mylist[1:2] = ['kiwi', 'mango']
print(mylist[2]) # Résultat : mango
```

```
# 4. Insertion d'un élément en début de liste
mylist = ['apple', 'banana', 'cherry']
mylist.insert(__index: 0, __object: 'orange')
print(mylist[1]) # Résultat : apple
# 5. Ajout avec append
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
print(fruits)
# 6. Insertion à la seconde position
fruits = ["apple", "banana", "cherry"]
fruits.insert(__index: 1, __object: "lemon")
print(fruits)
```

```
banana
['kiwi', 'banana', 'cherry']
mango
apple
['apple', 'banana', 'cherry', 'orange']
['apple', 'lemon', 'banana', 'cherry']
```

Exercise 5:

```
# 1. Suppression d'un élément avec remove
Semaine = ['lun', 'mar', 'mer', 'jeu', 'ven', 'sam', 'dim']
Semaine.remove('mer')
print(Semaine)

# 2. Suppression avec pop()
mylist = ['apple', 'banana', 'cherry']
mylist.pop(1)
print(mylist) # Résultat : ['apple', 'cherry']

# 3. Vider une liste
fruits = ['apple', 'banana', 'cherry']
fruits.clear() # On peut aussi utiliser del fruits[:]
print(fruits) # Résultat : []
```



```
['lun', 'mar', 'jeu', 'ven', 'sam', 'dim']
['apple', 'cherry']
[]
```

TP3 en Python:

Exercice 1:

```
exer1tp3.py > ...
1  nombres=[4,8,15,16,23,42]
2
3  #utiliser afflist()
4  def afflist(liste):
5      for element in liste:
6          print(element)
7
8  #appel à fonction
9  afflist(nombres)
```



```
PS C:\Users\EL HAMRI\Documents\vscode exercice> &
exercice/exer1tp3.py"
4
8
15
16
23
42
PS C:\Users\EL HAMRI\Documents\vscode exercice>
```

Exercice 2:

```
exerc2tp3.py > ...
1  #1
2  def somme_liste(liste):
3      return sum(liste)
4
5
6  def moyenne_liste(liste):
7      return somme_liste(liste)/len(liste)
8
9  #2
10 nombres=[1,2,3,4,5]
11
12 somme=somme_liste(nombres)
13 moyenne=moyenne_liste(nombres)
14
15 #3
16 print(f'somme est: {somme}')
17 print(f'moyenne est: {moyenne}')
```



```
exercice/exer2tp3.py"
somme est: 15
moyenne est: 3.0
PS C:\Users\EL HAMRI\Documents\vscode exercice>
```

Exercice 3:

```
exer3tp3.py > ...
1  #1
2  v def extrair_pairs(liste):
3      pairs=[]
4  v      for i in liste:
5  v          if i % 2 ==0:
6              pairs.append(i)
7
8      return pairs
9
0
1
2  #2
3  nombres=[1,2,3,4,5,36,17,89]
4
5  nombres_pairs=extrair_pairs(nombres)
6
7
8
9  #3
0  print(f'nombres pairs sont : {nombres_pairs}')
1
```



```
exercice/exer3tp3.py"
nombres pairs sont : [2, 4, 36]
PS C:\Users\EL HAMRI\Documents\vscode exercice>
```

Exercise 4:

```
def element_existe(liste, element):  
    if element in liste:  
        return True  
    else:  
        return False  
  
nombres = [10, 15, 20, 25, 30, 35, 40]
```

```
nombres = [10, 15, 20, 25, 30, 35, 40]  
  
print(element_existe(nombres, element: 15))  
print(element_existe(nombres, element: 50))
```



```
True  
False
```

Exercise 5:

```
exer5tp3.py > ...
1  #1
2  def list_carre(liste):
3      carre=[]
4      for i in liste:
5          |
6          |         carre.append(i**2)
7      return carre
8
9
10 |
11 |
12 #2
13 nombres=[1,2,3]
14
15 nombres_carre=list_carre(nombres)
16
17
18 #3
19 print(f'nombres carre sont : {nombres_carre}')
20
21
```



```
exercice/exer5tp3.py"
nombres carre sont : [1, 4, 9]
PS C:\Users\EL HAMRI\Documents\vscode ex
```


Exercice 6:

```
exer6tp3.py > ...
1  #1
2  def trouv_min_max(liste):
3
4      return (min(liste),max(liste))
5
6
7
8  #2
9  nombres=[1,2,3,56,71,5]
0
1  result=trouv_min_max(nombres)
2
3  |
4
5  #3
6  print(f'min et max sont : {result}')
7
```



```
exercice/exer6tp3.py"
min et max sont : (1, 71)
PS C:\Users\EL HAMRI\Documents\vscode exercice>
```

Exercice 7:

```
def fusionner_et_trier(liste1, liste2): 1 usage
    liste_fusionnee = liste1 + liste2
    # Trier la liste fusionnée
    liste_fusionnee.sort()
    return liste_fusionnee
nombres = [10, 25, 3, 0, 18]
autres_nombres = [7, 11, 19, 24]

print("Liste fusionnée et triée :", fusionner_et_trier(nombres, autres_nombres))
```



```
Liste fusionnée et triée : [0, 3, 7, 10, 11, 18, 19, 24, 25]
```

Exercice 8:

```
def est_palindrome(mot): 1 usage
    # On compare le mot avec sa version inversée
    return mot == mot[::-1]

mots = ["radar", "python", "level"]

for mot in mots:
    if est_palindrome(mot):
        print(f"'{mot}' est un palindrome.")
    else:
        print(f"'{mot}' n'est pas un palindrome.")
```



```
'radar' est un palindrome.
'python' n'est pas un palindrome.
'level' est un palindrome.
```

TP4 en Python:

Exercice 1:


```
exer1td4.py > chercher
1 dict={}
2 def ajoute():
3     nom=input('entrer un nom')
4     age=int(input('entrer l age'))
5     dict[nom]=age
6 a=int(input("entrer le nombre des valeur"))
7 for i in range(a):
8     ajoute()
9     print(dict)
10
11 def chercher():
12     nom=input("entrer un nom")
13     if nom in dict:
14         print(dict[nom])
15     else:
16         print('le nom nexist pes')
17 chercher()
18 def supprimer():
19     nom=input('nom a supprimer:')
20     if nom in dict:
21         del dict[nom]
22         print(f'{nom} a ete supprimer')
23     else:
24         print(f'{nom} n est pas dans le dictionnaire')
25 supprimer()
26
27
```



```
exercice/exer1td4.py"
entrer le nombre des valeur2
entrer un nomsara
entrer l age21
{'sara': 21}
entrer un nom
```

Exercise 2:

```
exercice2td4.py > ...
1 dict1={"g":4,"a":8,}
2 dict2={"c":7,"d":7}
3 dict3={}
4 def fusionner_diconnaire(dict1,dict2):
5     dict3.update(dict1)
6     dict3.update(dict2)
7     return dict3
8 fusion=fusionner_diconnaire(dict1,dict2)
9 print(f'fusion des deux dictionnaire est:{fusion}')
10
11 def trie_alphab(d1):
12     return{key:dict1[key]for key in sorted(dict1)}
13 def fusion3(dict1,dict2):
14     for key in sorted(dict1):
15         trier=trie_alphab(dict1)
16         print(f"les cles trier par ordre :{trier}")
17
18
19 fusion(dict1,dict2)
```

 `exercice/exercice2td4.py"`
fusion des deux dictionnaire est:{'g': 4, 'a': 8, 'c': 7, 'd': 7}

Exercice 3:

1/Créez un fichier texte contenant une liste de noms et d'âges (un par ligne, format `Nom,Âge`):

```
≡ Nom_Age.txt  
1 sara,22meriem,23jamila,21
```

2 et 3/Écrivez un programme qui lit ce fichier, stocke les données dans un dictionnaire, puis affiche ce dictionnaire:

```
"""  
donnees=[  
    "sara,22"  
    "meriem,23"  
    "jamila,21"  
]  
with open('Nom_Age.txt','w') as file:  
    for ligne in donnees:  
        file.write(ligne+"")
```

Exercise 4:

```
exer4td4.py > ...
1  with open("note-etudiant.txt","w") as file:
2      file.write("sara,20\n")
3      file.write("ayoub,10\n")
4      file.write("ilyasse,15\n")
5  with open("note-etudiant.txt","r") as file:
6      Notes=[]
7      Noms=[]
8      for ligne in file:
9          Nom,Note=ligne.strip().split(",")
10         Note=int(Note)
11         Noms.append(Nom)
12         Notes.append(Note)
13         Moyenne=sum(Notes)/len(Notes)
14 print(f"la note moyenne est:{Moyenne :2f}")
15 print("\netudiant avec une note sup à la moyenne :")
16 for nom,note in zip(Noms,Notes):
17     if note>Moyenne:
18         print(Nom)
19
20
21
22
23
```



```
exercice/exer4td4.py"
la note moyenne est:15.000000

etudiant avec une note sup à la moyenne :
ilyasse
PS C:\Users\EL HAMRI\Documents\vscode exercice>
```

Exercice 5:

```
exer5td4.py > ...
1 class Etudiant:
2     def __init__(self,nom,age,note):
3         self.nom=nom
4         self.age=age
5         self.note=note
6     def afficher_info(self):
7         print(f"le nom de l'etudiant est : {self.nom}, son age est : {self.age} et sa note : {self.note}")
8     @staticmethod
9     def moyenne(L):
10        somme=0
11        for ele in L:
12            somme+=ele.note
13        return somme /len(L)
14 et1=Etudiant("sara",20,17)
15 et2=Etudiant("ayoub",23,20)
16 et1.afficher_info()
17 et2.afficher_info()
18 etudiants=[et1,et2]
19 print("la moyenne est:",Etudiant.moyenne( etudiants))
```



```
exercice/exer5td4.py"
le nom de l'etudiant est : sara, son age est : 20 et sa note : 17
le nom de l'etudiant est : ayoub, son age est : 23 et sa note : 20
la moyenne est: 8.5
PS C:\Users\EL HAMRI\Documents\vscode exercice> |
```

Exercice 6:

```
exer6td4.py > ...
1
2 class CarnetAdresses:
3     def __init__(self):
4         self.contacts={}
5     def ajouter_contact(self,nom,email,telephone):
6         self.contacts[nom]={"email":email, "telephone":telephone}
7     def supprimer_contact(self,nom):
8         self.contacts.pop(nom,None)
9     def rechercher_contacts(self,nom):
10        return self.contacts.get(nom,"contact non trouvé")
11    def afficher_contacts(self):
12        return self.contacts
13    def record_file(self,fichier):
14        with open(fichier,"w",newline="") as f:
15            for nom,info in self.contacts.items():
16                f.write(f"{nom},{info['email']},{telephone}\n")
17
18    carnet= CarnetAdresses()
19    carnet.ajouter_contact("adam","adm@gmail.com","09876456")
20    carnet.ajouter_contact("ali","ali1@gmail.com","123456")
21    carnet.ajouter_contact("mohammed","mhm@gmail.com","78932132")
22    print(carnet.rechercher_contacts("adam"))
23    print("\n",carnet.afficher_contacts())
24    carnet.supprimer_contact("ahmed")
25    print("\n",carnet.afficher_contacts())
26    carnet.record_file("contacats.txt")
27
```



```
Les informations de 'adam' sont ajoutées.
Les informations de 'ali' sont ajoutées.
Les informations de 'mohmed' sont ajoutées.
adam: Email: adw@gmail.com, Téléphone: 09876456
ali: Email: ali1@gmail.com, Téléphone: 123456
mohmed: Email: mhmd@gmail.com, Téléphone: 78932132
Le contact 'adam' est supprimé.
Contact trouvé : ali, Email: ali1@gmail.com, Téléphone: 123456
Contact du adam, est introuvable !
Les contacts ont été sauvegardés dans 'carnet_adresses.txt'.
```


Exercice 7:

exerc7tp4.py > bibliotheque > all_books

```
1 class livre:
2     def __init__(self,titre,auteur,statut):
3         self.titre=titre
4         self.auteur=auteur
5         self.statut='disponible'
6     def __repr__(self):
7         return f"titre : {self.titre}\n auteur : { self.auteur} \n statut : {self.statut}"
8 instance=livre("mom",'auteur1','emprunte')
9 print( instance)
```

```
class bibliotheque:
    def __init__(self):
        self.livres=[]
    def ajouter_livre(self,titre):
        self.livres.append(livre)
    def all_books(self):
        return self.livres
    def emprunter_livre(self,titre):
        for L in self.livres:
            if L.titre==titre:
                if L.statut=="disponible":
                    L.statut=="emprunte"
            else:
                print("livre nom disponible")
    def rendre_livre(self,livre):
        for livre in self.livres:
            if livre.statut=="emprunte":
                livre.statut=="disponible"
```

```
livre=livre("mom","mooh","disponible")
les_livres= bibliotheque()
print(les_livres.all_books(livre))
```

TP5 en Python:

Pour ce dernier tp j'ai fait les exercices sur jupyter en utilisant la bibliothèque numpy.

Exercice 1:

```
#exer1
import numpy as np
#Création d'un tableau 1D contenant les nombres de 0 à 9
tableau_1D = np.arange(10)
print("Tableau 1D :")
print(tableau_1D)

# Création d'un tableau 2D de dimensions (3, 3) avec des nombres aléatoires entre 0 et 1
tableau_2D = np.random.rand(3, 3)
print("\nTableau 2D (3x3) avec des nombres aléatoires entre 0 et 1 :")
print(tableau_2D)

# Création d'un tableau 3D de dimensions (2, 3, 3) avec des nombres aléatoires entre 0 et 1
tableau_3D = np.random.rand(2, 3, 3)
print("\nTableau 3D (2x3x3) avec des nombres aléatoires entre 0 et 1 :")
print(tableau_3D)
```



```
Tableau 1D :
[0 1 2 3 4 5 6 7 8 9]

Tableau 2D (3x3) avec des nombres aléatoires entre 0 et 1 :
[[0.31945333 0.94468086 0.70917036]
 [0.93524093 0.663958 0.96999094]
 [0.67648847 0.68635071 0.35679117]]

Tableau 3D (2x3x3) avec des nombres aléatoires entre 0 et 1 :
[[[0.81812345 0.33189566 0.33986829]
  [0.19106176 0.4901796 0.50717928]
  [0.6303391 0.99292039 0.42007327]]
 [[0.35918567 0.78775712 0.04449107]
  [0.92892258 0.32973562 0.89420658]
  [0.11037249 0.98015141 0.3812381 ]]]
```

Exercice 2:

```
#exer2
import numpy as np

# Création de deux tableaux 1D de longueur 5
tableau_1 = np.array([1, 2, 3, 4, 5])
tableau_2 = np.array([5, 4, 3, 2, 1])

# Addition élément par élément
addition = tableau_1 + tableau_2
print("Addition élément par élément :", addition)

# Soustraction élément par élément
soustraction = tableau_1 - tableau_2
print("Soustraction élément par élément :", soustraction)

# Multiplication élément par élément
multiplication = tableau_1 * tableau_2
print("Multiplication élément par élément :", multiplication)

# Division élément par élément
division = tableau_1 / tableau_2
print("Division élément par élément :", division)
```



```
Addition élément par élément : [6 6 6 6 6]
Soustraction élément par élément : [-4 -2  0  2  4]
Multiplication élément par élément : [5 8 9 8 5]
Division élément par élément : [0.2 0.5 1.  2.  5. ]
```

```
# Création d'un tableau de valeurs entre 0 et  $\pi$ 
x = np.linspace(0, np.pi, 5)

# Application de np.sin, np.cos, et np.exp
sin_x = np.sin(x)
cos_x = np.cos(x)
exp_x = np.exp(x)

print("\nTableau de valeurs entre 0 et  $\pi$  :", x)
print("Sinus des valeurs :", sin_x)
print("Cosinus des valeurs :", cos_x)
print("Exponentielle des valeurs :", exp_x)
```



```
Tableau de valeurs entre 0 et  $\pi$  : [0.          0.78539816 1.57079633 2.35619449 3.14159265]
Sinus des valeurs : [0.00000000e+00 7.07106781e-01 1.00000000e+00 7.07106781e-01
1.22464680e-16]
Cosinus des valeurs : [ 1.00000000e+00 7.07106781e-01 6.12323400e-17 -7.07106781e-01
-1.00000000e+00]
Exponentielle des valeurs : [ 1.          2.19328005 4.81047738 10.55072407 23.14069263]
```

```
# Création d'un tableau d'entiers de longueur 10
tableau_entiers = np.arange(10)

# Sélectionner les éléments pairs
elements_pairs = tableau_entiers[tableau_entiers % 2 == 0]
print("\nÉléments pairs du tableau :", elements_pairs)

# Remplacer les éléments impairs par -1
tableau_modifie = np.where(tableau_entiers % 2 == 1, -1, tableau_entiers)
print("Tableau avec les impairs remplacés par -1 :", tableau_modifie)
```



```
Éléments pairs du tableau : [0 2 4 6 8]
Tableau avec les impairs remplacés par -1 : [ 0 -1  2 -1  4 -1  6 -1  8 -1]
```

Exercice 3:

1/ Créez un tableau de 12 éléments:

```
#exer3
import numpy as np

# Créer un tableau de 12 éléments
tableau_1D = np.arange(12) # Tableau allant de 0 à 11
print("\nTableau 1D initial :\n", tableau_1D)

# Transformer le tableau 1D en un tableau 2D de dimensions (3, 4)
tableau_2D = tableau_1D.reshape(3, 4)
print("\nTableau 2D (3x4) :\n", tableau_2D)

# Transformer le tableau 2D en un tableau 3D de dimensions (2, 2, 3)
tableau_3D = tableau_1D.reshape(2, 2, 3)
print("\nTableau 3D (2x2x3) :\n", tableau_3D)
```



```
Tableau 1D initial :
[ 0  1  2  3  4  5  6  7  8  9 10 11]

Tableau 2D (3x4) :
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

Tableau 3D (2x2x3) :
[[[ 0  1  2]
  [ 3  4  5]]
 [[ 6  7  8]
  [ 9 10 11]]]
```

2/ Transposez le tableau 2D obtenu précédemment et appliquez `np.swapaxes` et observez les résultats.

```
# Transposer le tableau 2D
tableau_2D_transpose = tableau_2D.T
print("\nTableau 2D transposé :\n", tableau_2D_transpose)
```

Tableau 2D transposé :

```
[[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
```

```
# Appliquer np.swapaxes pour échanger les axes 0 et 1 du tableau 2D
tableau_2D_swapaxes = np.swapaxes(tableau_2D, 0, 1)
print("\nTableau 2D après np.swapaxes (axes 0 et 1 échangés) :\n", tableau_2D_swapaxes)
```



Tableau 2D après np.swapaxes (axes 0 et 1 échangés) :

```
[[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
```

3/Créez deux tableaux 2D de dimensions (2, 3):

```
# Créer deux tableaux 2D de dimensions (2, 3)
tableau_2D_1 = np.array([[1, 2, 3], [4, 5, 6]])
tableau_2D_2 = np.array([[7, 8, 9], [10, 11, 12]])

print("\nTableau 2D 1 :\n", tableau_2D_1)
print("\nTableau 2D 2 :\n", tableau_2D_2)

# Concaténer les tableaux verticalement (axis 0)
concat_vertical = np.vstack((tableau_2D_1, tableau_2D_2))
print("\nConcaténation verticale (axis 0) :\n", concat_vertical)

# Concaténer les tableaux horizontalement (axis 1)
concat_horizontal = np.hstack((tableau_2D_1, tableau_2D_2))
print("\nConcaténation horizontale (axis 1) :\n", concat_horizontal)

# Diviser le tableau concaténé verticalement en sous-tableaux (2 sous-tableaux de 2x3)
sous_tableaux_vertical = np.split(concat_vertical, 2)
print("\nSous-tableaux après division verticale :\n", sous_tableaux_vertical)

# Diviser le tableau concaténé horizontalement en sous-tableaux (2 sous-tableaux de 2x3)
sous_tableaux_horizontal = np.split(concat_horizontal, 2, axis=1)
print("\nSous-tableaux après division horizontale :\n", sous_tableaux_horizontal)
```

Tableau 2D 1 :

```
[[1 2 3]
 [4 5 6]]
```

Tableau 2D 2 :

```
[[ 7  8  9]
 [10 11 12]]
```

Concaténation verticale (axis 0) :

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

Concaténation horizontale (axis 1) :

```
[[ 1  2  3  7  8  9]
 [ 4  5  6 10 11 12]]
```

Sous-tableaux après division verticale :

```
[array([[1, 2, 3],
        [4, 5, 6]]), array([[ 7,  8,  9],
        [10, 11, 12]])]
```

Sous-tableaux après division horizontale :

```
[array([[1, 2, 3],
        [4, 5, 6]]), array([[ 7,  8,  9],
        [10, 11, 12]])]
```



Exercice 4:

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:

```
#exer4
import numpy as np

# Créer un tableau aléatoire de dimensions (5, 5)
tableau_5x5 = np.random.rand(5, 5)
print("Tableau aléatoire 5x5 :\n", tableau_5x5)

# Calculer la moyenne par ligne et par colonne
moyenne_par_ligne = np.mean(tableau_5x5, axis=1)
moyenne_par_colonne = np.mean(tableau_5x5, axis=0)
print("\nMoyenne par ligne :", moyenne_par_ligne)
print("Moyenne par colonne :", moyenne_par_colonne)
```

```
# Calculer l'écart-type par ligne et par colonne
ecart_type_par_ligne = np.std(tableau_5x5, axis=1)
ecart_type_par_colonne = np.std(tableau_5x5, axis=0)
print("\nÉcart-type par ligne :", ecart_type_par_ligne)
print("Écart-type par colonne :", ecart_type_par_colonne)

# Calculer le minimum par ligne et par colonne
minimum_par_ligne = np.min(tableau_5x5, axis=1)
minimum_par_colonne = np.min(tableau_5x5, axis=0)
print("\nMinimum par ligne :", minimum_par_ligne)
print("Minimum par colonne :", minimum_par_colonne)

# Calculer le maximum par ligne et par colonne
maximum_par_ligne = np.max(tableau_5x5, axis=1)
maximum_par_colonne = np.max(tableau_5x5, axis=0)
print("\nMaximum par ligne :", maximum_par_ligne)
print("Maximum par colonne :", maximum_par_colonne)
```



Tableau aléatoire 5x5 :

```
[[0.79325068 0.72995842 0.27450231 0.13384305 0.20555102]
 [0.49006239 0.67845254 0.9599059 0.47119453 0.85217267]
 [0.0942015 0.92858072 0.70624909 0.18335128 0.58003569]
 [0.54546905 0.337901 0.12477064 0.92843469 0.99658324]
 [0.80824185 0.5471248 0.35697532 0.70897285 0.2774966 ]]
```

Moyenne par ligne : [0.4274211 0.69035761 0.49848366 0.58663172 0.53976228]

Moyenne par colonne : [0.54624509 0.6444035 0.48448065 0.48515928 0.58236784]

Écart-type par ligne : [0.27718538 0.19345943 0.31545306 0.33519144 0.20147559]

Écart-type par colonne : [0.25967463 0.19628683 0.30494269 0.30374061 0.3096165]

Minimum par ligne : [0.13384305 0.47119453 0.0942015 0.12477064 0.2774966]

Minimum par colonne : [0.0942015 0.337901 0.12477064 0.13384305 0.20555102]

Maximum par ligne : [0.79325068 0.9599059 0.92858072 0.99658324 0.80824185]

Maximum par colonne : [0.80824185 0.92858072 0.9599059 0.92843469 0.99658324]

2/C Créez un tableau de nombres aléatoires, triez-le dans l'ordre croissant et trouvez l'indice de l'élément maximum:

```
# Créer un tableau de nombres aléatoires de taille 10
tableau_aleatoire = np.random.rand(10)
print("\nTableau aléatoire :", tableau_aleatoire)

# Trier le tableau dans l'ordre croissant
tableau_trie = np.sort(tableau_aleatoire)
print("\nTableau trié dans l'ordre croissant :", tableau_trie)

# Trouver l'indice de l'élément maximum
indice_max = np.argmax(tableau_aleatoire)
print("\nIndice de l'élément maximum :", indice_max)

# Trouver l'élément maximum
element_max = tableau_aleatoire[indice_max]
print("Élément maximum :", element_max)
```



Tableau aléatoire : [0.73225948 0.09916418 0.83396591 0.45090143 0.18256537 0.28898181
0.3479816 0.40203274 0.31847398 0.1118542]

Tableau trié dans l'ordre croissant : [0.09916418 0.1118542 0.18256537 0.28898181 0.31847398 0.3479816
0.40203274 0.45090143 0.73225948 0.83396591]

Indice de l'élément maximum : 2

Élément maximum : 0.8339659065170636

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

```
# Créer un tableau 1D de longueur 4
tableau_1D = np.array([1, 2, 3, 4])
print("Tableau 1D :\n", tableau_1D)

# Créer un tableau 2D de dimensions (3, 4)
tableau_2D = np.array([[5, 6, 7, 8],
                        [9, 10, 11, 12],
                        [13, 14, 15, 16]])
print("\nTableau 2D :\n", tableau_2D)

# Multiplier les tableaux en utilisant le broadcasting
resultat = tableau_2D * tableau_1D
print("\nRésultat de la multiplication (broadcasting) :\n", resultat)
```



```
Tableau 1D :
[1 2 3 4]

Tableau 2D :
[[ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

Résultat de la multiplication (broadcasting) :
[[ 5 12 21 32]
 [ 9 20 33 48]
 [13 28 45 64]]
```

Exercice 6:(suite du tp5)

```
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}")
```

Pyt



Total des ventes : P1 = 6258, P2 = 6117, P3 = 7187

Moyenne des ventes : P1 = 521.50, P2 = 509.75, P3 = 598.92

Mois avec ventes maximales : P1 = Mois 6, P2 = Mois 3, P3 = Mois 12

CONCLUSION :

À travers la réalisation de cinq travaux pratiques en Python, j'ai pu approfondir ma maîtrise des concepts fondamentaux du langage, tels que la gestion des variables, les structures de contrôle et les fonctions. Ces exercices ont démontré la simplicité et la flexibilité de Python, tout en me permettant de développer une meilleure compréhension de l'optimisation du code. Python s'est révélé être un langage puissant, adapté à de nombreux domaines, et ce travail m'a donné envie de poursuivre son apprentissage pour des projets futurs.