



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)
```



```
PS C:\Users\EL HAMRI\Documents\vscode exercice> ^C

PS C:\Users\EL HAMRI\Documents\vscode exercice> & "C:/Users/EL HAMRI/AppData/Local/P cice/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 > ② coef
    """calcul et affiche la moyenne des notes"""
    math=input("entrer math result")
    physique=input("entrer physique result")
    svt=input("entrer svt result")
    anglais=input("entrer anglais result")
    coef-math=input("entrer coef-math")
```

2/calcul du budget des achats :

```
coef-physique=input("entrer coef-physique")

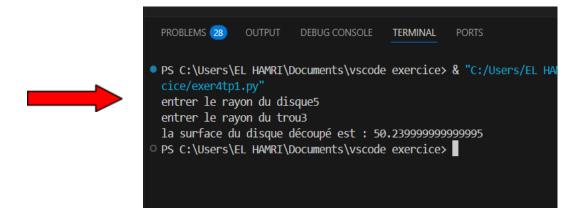
coef-svt=input("entrer coef-svt")

coef-anglais=input("entrer coef-anglais")

total-coef=int(coef-math)+int(coeft-physique)+int(coef-svt)+int(coef-anglais)

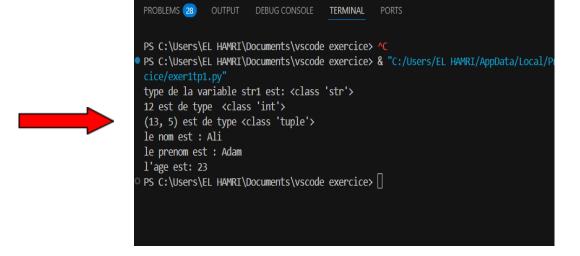
matier avec coeff-float(math)*int(coef-math)+float(physique)*int(coef-physique)+float(svt)*int(coef-svt)+float(anglais)*int(coef-anglais)

moyenne=matiere avec coef/total coef
```



Exercice 3:

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



Exercice 4:

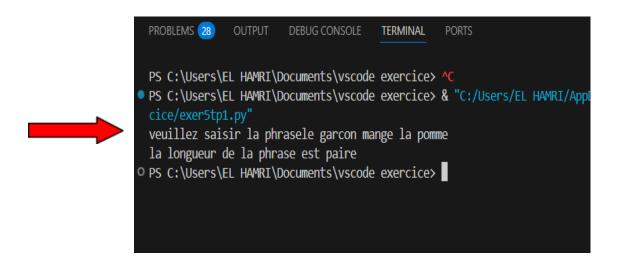
```
rg=float(input("entrer le rayon du disque"))
rp=float(input("entrer le rayon du trou"))

vif rg>rp:
S=3.14*rg**2-3.14*rp**2
print("la surface du disque découpé est :", S)

velse:
print("vous ne pouvez pas découper le disque")
```

Exercice 5:

```
exer5tp1.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")
```



TP2 en Python:

Exercice 1:

1/Définition et affichage d'une liste :

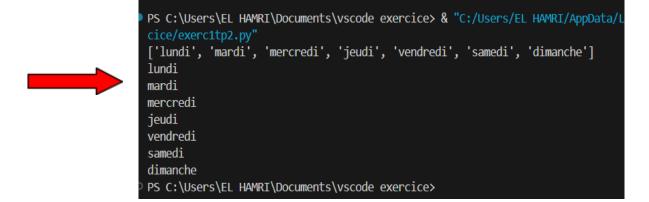
```
exercitp2.py / ...

semaine=["lundi","mardi","mercredi","jeudi","vendredi","samedi","dimanche"]

print(semaine)

for jour in semaine:

print(jour)
```



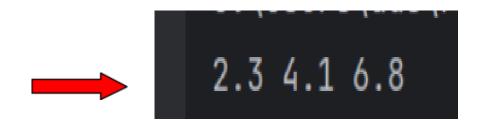
2/Affichage de la liste couleurs :

```
"""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])
```



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(hislist))
# 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 Se 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 > ...
     matieres =['anglais','physiques','maths','svt']
      #affichage
      print(matieres)
      for m in matieres:
          print(m)
      #ajout de matieres
     matieres.append('histoires')
     #ajout en utilisant insert()
     matieres.insert(2, 'geographie')
      print (matieres)
      #utilisant operateur []
      print(matieres[:4])
      print(matieres[-3:])
      print(matieres[2:5])
19
```

```
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']
```

Exercice 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 > ...
    nombres=[4,8,15,16,23,42]

#utiliser afflist()
    def afflist(liste):
        for element in liste:
            print(element)

#appel à fonction

afflist(nombres)
```

Exercice 2:

```
exer2tp3.py > ...
      #1
      def somme liste(liste):
          return sum(liste)
      def moyenne liste(liste):
          return somme_liste(liste)/len(liste)
      #2
      nombres=[1,2,3,4,5]
11
      somme=somme liste(nombres)
12
      moyenne=moyenne liste(nombres)
13
15
      #3
      print(f'somme est: {somme}')
      print(f'moyenne est: {moyenne}')
17
```

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

Exercice 3:



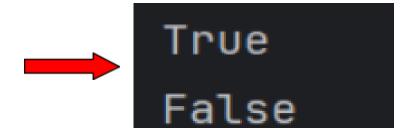
exercice/exer3tp3.py"
nombres pairs sont : [2, 4, 36]
PS C:\Users\EL HAMRI\Documents\vscode exercice>

Exercice 4:

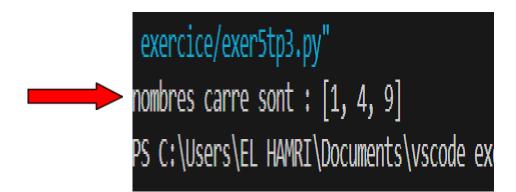
```
def element_existe(liste, element):
    if element in liste:
        return True
    else:
        return False

nombres = [10, 15, 20, 25, 30, 35, 4]
```

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



Exercice 5:



Exercice 6:

```
exercice/exer6tp3.py"

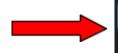
min et max sont : (1, 71)

PS C:\Users\EL HAMRI\Documents\vscode exercice>
```

Exercice 7:

```
def fusionner_et_trier(liste1, liste2): 1usage
    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.")
```

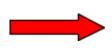


'radar' est un palindrome.
'python' n'est pas un palindrome.
'level' est un palindrome.

TP4 en Python:

Exercice 1:

```
exer1td4.py > 😭 chercher
      dict={}
      def ajoute():
         nom=input('entrer un nom')
         age=int(input('entrer 1 age'))
         dict[nom]=age
      a=int(input("entrer le nombre des valeur"))
      for i in range(a):
         ajoute()
         print(dict)
      def chercher():
        nom=input("entrer un nom")
        if nom in dict:
          print(dict[nom])
        else:
          print('le nom nexist pes')
16
      chercher()
      def supprimer():
        nom=input('nom a supprimer:')
        if nom in dict:
          del dict[nom]
          print(f'{nom} a ete supprimer')
          print(f'{nom} n est pas dans le dictionnaire')
      supprimer()
```



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

Exercice 2:

```
exercice2td4.py > ...
      dict1={"g":4,"a":8,}
      dict2={"c":7,"d":7}
      dict3={}
      def fusionner diconnaire(dict1,dict2):
          dict3.update(dict1)
          dict3.update(dict2)
          return dict3
      fusion=fusionner diconnaire(dict1, dict2)
      print(f'fusion des deux dictionnaire est:{fusion}')
      def trie alphab(d1):
11
        return{key:dict1[key]for key in sorted(dict1)}
12
      def fusion3(dict1,dict2):
13
         for key in sorted(dict1):
          trier=trie alphab(dict1)
15
         print(f"les cles trier par ordre :{trier}")
17
      fusion(dict1, dict2)
19
```

```
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+"")
```

Exercice 4:

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



```
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:
 self.nom=nom
         self.age=age
         self.note=note
        def afficher info(self):
          print(f"le nom de l'etudiant est : {self.nom}, son age est : {self.age} et sa note : {self.note}")
         @staticmethod
        def moyenne(L):
         somme=0
         for ele in L:
           somme+=ele.note
           return somme /len(L)
    et1=Etudiant("sara",20,17)
     et2=Etudiant("ayoub",23,20)
     et1.afficher_info()
    et2.afficher info()
     etudiants=[et1,et2]
18
     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 > ...
      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 contacts(self,nom):
              return self.contacts.get(nom, "contact non trouvé")
          def afficher contacts(self):
11
              return self.contacts
12
          def record file(self, fichier):
            with open(fichier, "w", newline="") as f:
              for nom,info in self.contacts.items():
                 f.write(f"{nom},{info['email']},{telephone}\n")
      carnet= CarnetAdresses()
      carnet.ajouter contact("adam", "adm@gmail.com", "09876456")
      carnet.ajouter_contact("ali","ali1@gmail.com","123456")
      carnet.ajouter contact("mohammed", "mhm@gmail.com", "78932132")
      print(carnet.rechercher contacts("adam"))
      print("\n",carnet.afficher contacts())
      carnet.supprimer contact("ahmed")
      print("\n", carnet.afficher contacts())
      carnet.record file("contacats.txt")
```

```
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'.
```

Les informations de 'adam' sont ajoutées.

Exercice 7:

```
exerc7tp4.py > % bibliotheque > % all_books

class livre:

def __init__(self,titre,auteur,statut):

self.titre=titre

self.auteur=auteur

self.statut='disponible'

def __repr__(self):

return f"titre : {self.titre}\n auteur : { self.auteur} \n statut : {self.statut}"

sinstence=livre("mom",'auteur1','emprunte')

print( instence)
```

```
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:

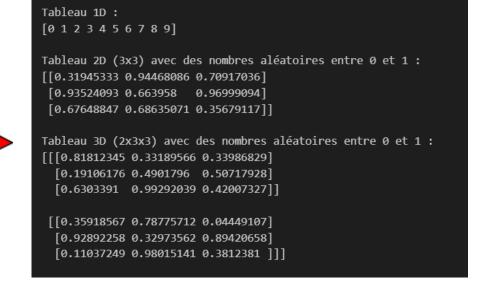
Pour ce dernier tp j'ai fait les exercices sur jupyter en utilisant la bibliothèque numpy.

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("\nTableau 3D (2x3x3) avec des nombres aléatoires entre 0 et 1 :")
print(tableau_3D)
```



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]

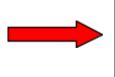
Multiplication élément par élément : [5 8 9 8 5]

Division élément par élément : [0.2 0.5 1. 2. 5.]
```

```
# Création d'un tableau de valeurs entre 0 et π
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 π :", x)
print("Sinus des valeurs :", sin_x)
print("Cosinus des valeurs :", cos_x)
print("Exponentielle des valeurs :", exp_x)
```



```
# 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("Tableau 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)
```

```
# 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)
```

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("\nMaximum 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]
```

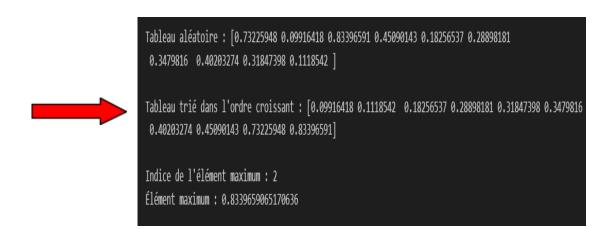
2/Créez un tableau de nombres aléatoires, triez-le dans l'ordre croissant et trouvez l'indice de l'élément maximum:

```
# 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)
```



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



```
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.