Royaume du Maroc
Université Mohammed Premier
École Supérieure De Technologie-Oujda
Département : Informatique
Filière : LP Informatique Décisionnelle

Analyse De Données

Compte Rendu Des Travaux Pratiques Sur Python Dans Google Colab 'TP1'

Réalisé par : **HNIOUA Abdessamad**

Encadré par : **M. Mounir Grari**

Année Universitaire : 2023/2024

Introduction:

Python est devenu un pilier incontournable dans le domaine de l'analyse de données et de la prise de décision. Sa popularité croissante repose sur sa simplicité syntaxique, sa polyvalence et sa richesse en bibliothèques spécialisées. En effet, Python offre une gamme d'outils et de modules spécialement conçus pour l'analyse, la manipulation et la visualisation des données, ce qui en fait un choix privilégié pour les professionnels et les chercheurs souhaitant extraire des insights significatifs à partir de volumes de données massifs. Cette introduction explore l'importance de Python dans le contexte de l'analyse de données et de la prise de décision, en mettant en lumière ses avantages et son impact significatif dans divers domaines d'application.

Avant de plonger dans le domaine complexe de l'analyse de données, il est essentiel de maîtriser les bases de la programmation en Python.

Exercice 1 «Lecture de code»

Lorsque vous voyez un message d'erreur en Python, vous pouvez consulter le traceback pour suivre la séquence d'appels de fonction jusqu'à l'endroit où l'erreur s'est produite, identifier le fichier et le numéro de ligne spécifiques où l'erreur est survenue, et comprendre le type d'exception qui a été levée.

1. On a effacé les lignes de sortie de l'interpréteur. Écrire la sortie attendue.

```
x = 8
x + 2
x ** 2   //x à la puissance 2
x * x
x = x + 1
x     // 9 Les instructions 2 à 4 effectuent des opérations arithmétiques, mais elles ne modifient pas la valeur de la variable x.
```

Exercice 2 «Opérations»

Les types de base : int, float, bool

isinstance(obj, type) -> bool permet de vérifier un type précis

```
type(2)
type(2.0)
type(2 == 2.0)
x=type(2) == int
y=isinstance(2==3, int)
```

```
print( type(2) , type(2.0) , x,y)
// la fonction type return le type de variable  exmple var 2 est
de type int
```

Exercice 3 « Opérations sur les entiers et les flottants »

1. Proposer une expression donnant le nombre de secondes écoulées entre le premier janvier de cette année à minuit et ce matin, à minuit.

```
from datetime import datetime
debut_annee = datetime(datetime.now().year, 1, 1, 0, 0, 0)
debut_jour  =  datetime(datetime.now().year,  datetime.now().month,
datetime.now().day, 0, 0, 0)
jour=debut_jour-debut_annee
seconds= jour * 24 * 60 * 60
seconds=jour.total_seconds();
print("les seconds " ,seconds)
```

Le module **datetime** fournit des classes pour manipuler les dates et les heures. Vous pouvez utiliser la classe datetime pour représenter des objets de date et d'heure.

2. Pierre, Paul et Jacques ont acheté par erreur 2 ** 11 gâteaux à la boulangerie. Ils les répartissent équitablement. Combien chacun en aura-t-il ? Combien en reste-t-il ?

```
nbGâteauxParBoulangerie = (2**11) // 3
ResteGâteaux = (2**11) % 3
print("Nombre des Gâteauxs Par Boulangeries:",nbGâteauxParBoulangerie)
print("Gâteaux Reste:",ResteGâteaux)
```

```
Nombre des Gâteauxs Par Boulangeries : 682
Gâteaux Reste : 2
```

La fonction **round()** en Python est utilisée pour arrondir un nombre à un certain nombre de décimales.

```
round(2.623335,3)   // 2.623
```

Exercice 4 « Opérations sur les booléens »

- Evaluation de les expressions booléennes suivantes :
  - not (1 == 2) //True
  - (1 == 2) or (2 ** 2 == 4) // False
  - (4 <= 3) or (5 > 2) // True

- x est un variable de type float. Proposer une expression booléenne permettant de vérifier que :
  1. x ∈ [1;9]

```
x=3
1 <= x <= 9
// True
```

  2. x ∈]−∞;0[U]2;3]

```
x=3
(x < 0) or (2 <= x <= 3)
//True
```

- Pour vérifier si un entier n est divisible par trois sans qu'il ne soit égal à zéro, vous pouvez utiliser l'expression booléenne suivante en Python :

```
n=1
n != 0 and n % 3 == 0
// False
```

- Pour déterminer si une année est bissextile selon les règles que vous avez mentionnées, vous pouvez utiliser la condition suivante en Python :

```
annee = 2023
annee % 4 == 0 and (annee % 100 != 0 or annee % 400 == 0)
```

Exercice 5 « Affectation»

Une liste peut contenir des éléments de types différents. Voici comment vous pouvez réaliser 4 affectations à 4 variables avec une liste contenant des éléments de types différents

```
a = "22dhs"
b = "Prix:"
c = a + b
d = 22
d = "22"
ma_liste = [1, "2", "trois"]
ma_liste = [1, "2", "trois"]
ma_liste = ["1", 2, "trois"]
ma_liste[2] = 9
ma_liste[2] = "DEUX"
e = ("b" == 22)
f = True
```

Exercice 6 « Échanger deux valeurs »

```
a = 3
b = 6
temp = b
b=a
```

```
a=temp
print("a=",a,"b=",b)
```

Exercice 7 « évaluer »

```
def func(a, b):
  return a + b
func(12, 10)
# 22
func("bonjour", "la famille")
# 'bonjourla famille'
func(12, "bonjour")
# TypeError: unsupported operand type(s) for +: 'int' and 'str'
func(10, func(5, 7))
# 22
```

Exercice 8 « définir »

```
def initiales(a, b):
  return a[0]+b[0]
initiales("Georges", "Garmin")
# GG
```

Améliorer la fonction pour qu'elle renvoie vrai si la première est dans la seconde ou la seconde est dans la première.

```
def contient2(a, b):
  return a in b or b in a
contient2("abdedefg", "def")
# True
```

Programmer est_pair qui prend un entier et renvoie vrai s'il est pair.

```
def est_pair(nb):
  return nb%2==0
est_pair(2)
# True
```

```
True
```

divMod est une fonction native qui prend deux entiers a et b et renvoie le quotient de a par b ainsi que le reste de cette division. Programmer cette fonction.

```
def divMod(a,b):
  div=a/b
```

```
  mod=a%b
  print("div:",round(div,2), "mod:",round(mod,2));
divMod(20,19)
```

```
div: 1.05 mod: 1
```

Exercice 9 « Effets de bord »

Conditions - if elif else

```
age = 23
if age > 18:
  majeur = True
else:
  majeur = False
print(majeur)
```

```
True
```

Exercice 10 « if elif else »

Écrire une fonction qui prend une note (nombre entre 0 et 20) et renvoie la mention associée.

```
def mention(note):
    if note >= 10:
        if 10 <= note < 12:
            print("La mention est passable")
        elif 12 <= note < 14:
            print("La mention est assez bien")
        elif 14 <= note < 16:
            print("La mention est bien")
        else:
            print("La mention est excellente")
    else:
        print("La note est inférieure à 10 ")

mention(12.3)
```

```
La mention est assez bien
```

Exercice 11 - Boucles non bornées : while

On a affecté à f la fonction mathématique f(x) = x2 +5x−2. Programmer une fonction seuil qui prend un nombre a et renvoie le premier entier n tel que f(n) > a.

```
def f(x):
    return x**2 + 5*x - 2
```

```
def seuil(a):
    n = 0
    while True:
        if f(n) > a:
            return n
        n += 1
print("seuil est :",seuil(10))
```

```
seuil est : 2
```

Programmer une fonction nb_annee_pour_doubler qui prend un taux d'intérêt t en pourcentage (positif) et renvoie le nombre d'années nécessaires pour doubler un capital placé à intérêts composés avec le taux t.

```
def nb_annee_pour_doubler(taux_interet):
    capital_initial = 1
    montant_double = capital_initial * 2
    annees = 0

    while capital_initial < montant_double:
        capital_initial *= (1 + taux_interet / 100)
        annees += 1
    return annees
taux = 5
print("Nombre d'années pour doubler le capital avec un taux de {}% :
{}".format(taux, nb_annee_pour_doubler(taux)))
```

```
Nombre d'années pour doubler le capital avec un taux de 5% : 15
```

Exercice 12 - Boucles bornées : for

```
nbPairs = []
nb = 1
while nb < 100:
    if nb % 2 == 0:
        nbPairs.append(nb)
    nb += 1
somme = 0
for elem in nbPairs:
    somme += elem
print("La somme des entiers pairs plus petits que 100 est :", somme)
```

```
La somme des entiers pairs plus petits que 100 est : 2450
```

Avec la fonction range() :

```
somme = 0
for nombre in range(2, 100, 2):
    somme += nombre

print("La somme des entiers pairs plus petits que 100 est :", somme)
```

```
La somme des entiers pairs plus petits que 100 est : 2450
```

Afficher la table complète de multiplication pour le nombre 5 :

```
nombre = 5
for i in range(11):
    print("{} * {} = {}".format(nombre, i, nombre * i))
```

```
5 * 0 = 0
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

Int (input('Entrez un nombre')) pour demande de saisir un nombre ;

```
nombre = int(input("Entrez le nombre pour lequel vous voulez afficher la table de multiplication : "))
for i in range(11):
    print("{} * {} = {}".format(nombre, i, nombre * i))
```

```
Entrez le nombre pour lequel vous voulez afficher la table de multiplication : 2
2 * 0 = 0
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

Afficher toutes les tables de multiplication de 1 à 10 avec une ligne de séparation entre chaque table :

```
for nombre in range(1, 11):
    print("Table de multiplication pour", nombre)
    for i in range(11):
        print("{} * {} = {}".format(nombre, i, nombre * i))
    print()
```

```
Table de multiplication pour 1
1 * 0 = 0
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10

Table de multiplication pour 2
2 * 0 = 0
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

```
Table de multiplication pour 3
3 * 0 = 0
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30

Table de multiplication pour 4
4 * 0 = 0
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
```

```
Table de multiplication pour 5
5 * 0 = 0
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

Table de multiplication pour 6
6 * 0 = 0
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60
```

```
Table de multiplication pour 7
7 * 0 = 0
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70

Table de multiplication pour 8
8 * 0 = 0
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80
```

```
Table de multiplication pour 9
9 * 0 = 0
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90

Table de multiplication pour 10
10 * 0 = 0
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
```

Exercice 13 - Un premier types complexe : list

1. Quels sont les premiers et derniers éléments de mes_carres ?

```
mes_carres = []
for i in range(100):
    if i % 2 == 0:
        mes_carres.append(i ** 2)
print("Premier élément de mes_carres:", mes_carres[0])
print("Dernier élément de mes_carres:", mes_carres[-1])
```

```
Premier élément de mes_carres: 0
Dernier élément de mes_carres: 9604
```

2. Comment accéder à la longueur de la liste mes_carres ? fonction len().

```
mes_carres = []
for i in range(100):
    if i % 2 == 0:
        mes_carres.append(i ** 2)
len(mes_carres)
```

```
50
```

3. Modifier le code pour déterminer les carrés des entiers divisibles par 3.

```
mes_carres = []
for i in range(100):
    if i % 3 == 0:
```

```
        mes_carres.append(i ** 2)
print("La taille des entiers divisibles par 3:", len(mes_carres))
print("Les carrés des entiers divisibles par 3:", mes_carres)
```

```
La taille des entiers divisibles par 3: 34
Les carrés des entiers divisibles par 3: [0, 9, 36, 81, 144, 225, 324, 441, 576, 729, 900, 1089, 1296, 1521, 1764, 2025,
```

4. Créer la liste de l'énoncé précédent sans utiliser if

```
mes_carres = []
for i in range(0,100,3):
        mes_carres.append(i ** 2)
print("La taille des entiers divisibles par 3:", len(mes_carres))
print("Les carrés des entiers divisibles par 3:", mes_carres)
```

```
La taille des entiers divisibles par 3: 34
Les carrés des entiers divisibles par 3: [0, 9, 36, 81, 144, 225, 324, 441, 576, 729, 900, 1089, 1296, 1521, 1764, 2025,
```

5. Robert commence un régime. Le mardi et le vendredi, il ne se nourrit plus que de fruits.

Écrire un programme qui affiche chaque jour de la semaine et le type d'alimentation de Robert :

Le lundi tu peux manger ce que tu veux,

Le mardi tu dois manger des fruits,

On utilisera une liste pour enregistrer les jours ["lundi",...]

```
jours_semaine = ["lundi", "mardi", "mercredi", "jeudi", "vendredi",
"samedi", "dimanche"]
for jour in jours_semaine:
    if jour == "mardi" or jour == "vendredi":
        print(f"Le {jour} tu dois manger des fruits.")
    else:
        print(f"Le {jour} tu peux manger ce que tu veux.")
```

```
Le lundi tu peux manger ce que tu veux.
Le mardi tu dois manger des fruits.
Le mercredi tu peux manger ce que tu veux.
Le jeudi tu peux manger ce que tu veux.
Le vendredi tu dois manger des fruits.
Le samedi tu peux manger ce que tu veux.
Le dimanche tu peux manger ce que tu veux.
```

Exercice 14 - Liste et boucle for

La lettre f est utilisée pour indiquer une "f-string" en Python. Une f-string est une fonctionnalité de formatage de chaîne de

caractères qui permet d'incorporer des variables et des expressions directement dans une chaîne en préfixant la chaîne avec la lettre

```
mes_enfants = ["Enfant1", "Enfant2", "Enfant3", "Enfant4", "Enfant5", "Enfant6", "Enfant7", "Enfant8", "Enfant9", "Enfant10", "Enfant11", "Enfant12"]

for i in range(13 - len(mes_enfants)):
    mes_enfants.append(f"Enfant{len(mes_enfants) + 1}")

print(mes_enfants)


for i in range(len(mes_enfants)):
    mes_enfants[i] = f"Squeezie {i + 1}"
print(mes_enfants)
```

```
['Enfant1', 'Enfant2', 'Enfant3', 'Enfant4', 'Enfant5', 'Enfant6', 'Enfant7', 'Enfant8', 'Enfant9', 'Enfant10', 'Enfant11', 'Enfant12', 'Enfant13']
['Squeezie 1', 'Squeezie 2', 'Squeezie 3', 'Squeezie 4', 'Squeezie 5', 'Squeezie 6', 'Squeezie 7', 'Squeezie 8', 'Squeezie 9', 'Squeezie 10', 'Squeezie 11', 'Squeezie 12', 'Squeezie 13']
```

On considère deux variables :

lettres = ['a', 'b', ..., 'z'] et voyelles = ['a', 'e', 'i', 'o', 'u', 'y'] :

Écrire un programme python qui affiche chaque lettre de l'alphabet avec un commentaire.

```
lettres = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
voyelles = ['a', 'e', 'i', 'o', 'u', 'y']

for i in lettres:
    if i in voyelles:
        print(f"{i} est une voyelle")
    else:
        print(f"{i} n'est pas une voyelle")
```

```
a est une voyelle
b n'est pas une voyelle
c n'est pas une voyelle
d n'est pas une voyelle
e est une voyelle
f n'est pas une voyelle
g n'est pas une voyelle
h n'est pas une voyelle
i est une voyelle
j n'est pas une voyelle
k n'est pas une voyelle
l n'est pas une voyelle
m n'est pas une voyelle
n n'est pas une voyelle
o est une voyelle
p n'est pas une voyelle
q n'est pas une voyelle
r n'est pas une voyelle
s n'est pas une voyelle
t n'est pas une voyelle
u est une voyelle
v n'est pas une voyelle
w n'est pas une voyelle
x n'est pas une voyelle
y est une voyelle
z n'est pas une voyelle
```

Écrire un programme affichant les titres des albums sortis une année paire. On créera deux listes : celle des années et celle des titres.

```
annees = [1963, 1963, 1964, 1964, 1965, 1965, 1966, 1967, 1968, 1969,
1969, 1970]
titres = ["Please Please Me", "With the Beatles", "A Hard Day's Night",
"Beatles for Sale", "Help!", "Rubber Soul",
            "Revolver", "Sgt. Pepper's Lonely Hearts Club Band", "The
Beatles", "Yellow Submarine", "Abbey Road", "Let It Be"]
for annee, titre in zip(annees, titres):
    if annee % 2 == 0:
            print("Les albums des Beatles sortis dans {} est :
{}".format(annee,titre))
```

Retour sur les albums des Beatles. On décide d'enregistrer les albums dans une liste de couples :

```
annees = [1963, 1963, 1964, 1964, 1965, 1965, 1966, 1967, 1968, 1969,
1969, 1970]
titres = ["Please Please Me", "With the Beatles", "A Hard Day's Night",
"Beatles for Sale", "Help!",
          "Rubber Soul", "Revolver", "Sgt. Pepper's Lonely Hearts Club
Band", "The Beatles", "Yellow Submarine", "Abbey Road", "Let It Be"]
albums = []
for annee, titre in zip(annees, titres):
    if annee % 2 == 0:
        albums.append((annee, titre))
for album in albums:
    print(album)
```

```
(1964, 'A Hard Day's Night')
(1964, 'Beatles for Sale')
(1966, 'Revolver')
(1968, 'The Beatles')
(1970, 'Let It Be')
Les albums des Beatles sortis dans 1964 est : A Hard Day's Night
Les albums des Beatles sortis dans 1964 est : Beatles for Sale
Les albums des Beatles sortis dans 1966 est : Revolver
Les albums des Beatles sortis dans 1968 est : The Beatles
Les albums des Beatles sortis dans 1970 est : Let It Be
```

Écrire une boucle qui affiche les années et titres des albums dont le titre contient la lettre “a”.

```
annees = [1963, 1963, 1964, 1964, 1965, 1965, 1966, 1967, 1968, 1969,
1969, 1970]
titres = ["Please Please Me", "With the Beatles", "A Hard Day’s Night",
"Beatles for Sale", "Help!",
          "Rubber Soul", "Revolver", "Sgt. Pepper’s Lonely Hearts Club
Band", "The Beatles", "Yellow Submarine", "Abbey Road", "Let It Be"]
albums = []
for annee, titre in zip(annees, titres):
        albums.append((annee, titre))
for annee , titre in albums:
  if ( "a" or "A" ) in titre:
      print("Le album des Beatles qui  contient la lettre “a” sortie
dans  {} est : {}".format(annee,titre))
```

```
Le album des Beatles qui  contient la lettre “a” sortie dans  1963 est : Please Please Me
Le album des Beatles qui  contient la lettre “a” sortie dans  1963 est : With the Beatles
Le album des Beatles qui  contient la lettre “a” sortie dans  1964 est : A Hard Day’s Night
Le album des Beatles qui  contient la lettre “a” sortie dans  1964 est : Beatles for Sale
Le album des Beatles qui  contient la lettre “a” sortie dans  1967 est : Sgt. Pepper’s Lonely Hearts Club Band
Le album des Beatles qui  contient la lettre “a” sortie dans  1968 est : The Beatles
Le album des Beatles qui  contient la lettre “a” sortie dans  1969 est : Yellow Submarine
Le album des Beatles qui  contient la lettre “a” sortie dans  1969 est : Abbey Road
```

Écrire une fonction qui prend une liste et renvoie sa longueur. On n’utilisera pas len

```
def TailleListe(maListe):
    taille = 0
    for element in maListe:
        taille += 1
    return taille
annees = [1963, 1963, 1964, 1964, 1965, 1965, 1966, 1967, 1968, 1969,
1969, 1970]
print("La taille de la liste est : ",TailleListe(annees))
```

```
La taille de la liste est :  12
```

Écrire une fonction index qui prend une liste l et un objet x et renvoie :

- le premier indice de x dans l
- -1 si x n’est pas dans l

```
def index(l, x):
    for i, element in enumerate(l):
        if element == x:
            return i
    return -1
liste = [1, 2, 3, 4, 5, 3, 6, 7, 8]
```

```
objet = 3
resultat = index(liste, objet)
print("Le premier indice de", objet, "dans la liste est :", resultat)
```

```
Le premier indice de 3 dans la liste est : 2
```

Écrire la fonction copie_pairs qui prend une liste d'entiers et renvoie la copie des entiers pairs de la liste.

Écrire une fonction qui prend une liste et renvoie une copie renversée de la liste.

```
def copie_pairs(liste):
    copie = []
    for nombre in liste:
        if nombre % 2 == 0:
            copie.append(nombre)
    return copie

def copie_renversee(liste):
    return liste[::-1]

liste_entiers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
copie_pairs_resultat = copie_pairs(liste_entiers)
copie_renversee_resultat = copie_renversee(liste_entiers)

print("Liste d'entiers originale :", liste_entiers)
print("Copie des entiers pairs :", copie_pairs_resultat)
print("Copie renversée de la liste originale :",
copie_renversee_resultat)
```

```
Liste d'entiers originale : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Copie des entiers pairs : [2, 4, 6, 8, 10]
Copie renversée de la liste originale : [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Exercice 15 - Compléments sur les fonctions et les listes

1- Écrire une fonction est_triee qui prend en paramètre une liste et renvoie un booléen vrai si la liste est triée.
2- Écrire une fonction concat qui prend deux listes l_1 et l_2 et renvoie une nouvelle liste formée des éléments de l_1 suivi des éléments de l_2

```
def est_triee(liste):
    for i in range(len(liste) - 1):
        if liste[i] > liste[i + 1]:
            return False
    return True

def concat(l_1, l_2):
    return l_1 + l_2

# Exemples d'utilisation :
liste_non_triee = [3, 1, 4, 1, 5, 9]
liste_triee = [1, 2, 3, 4, 5, 6]

print(est_triee(liste_non_triee))
print(est_triee(liste_triee))

l1 = [1, 2, 3]
l2 = [4, 5, 6]
print(concat(l1, l2))
```

```
False
True
[1, 2, 3, 4, 5, 6]
```

3- Cette fonction prend deux listes, supposées de même taille et renvoie une nouvelle liste contenant les sommes des éléments de chaque liste**.**

```
def ajoute_paire(l_1, l_2):
    l_3 = []
    for i in range(len(l_1)):
        somme = l_1[i] + l_2[i]
        l_3.append(somme)
    return l_3
resultat = ajoute_paire([1, 2, 3], [10, 12, 15])
print(resultat)
```

```
[11, 14, 18]
```

4- Le type de sortie de la fonction mystère est un nombre décimal, c'est-à-dire un flottant en Python.

Si la liste est vide, c'est-à-dire si l est une liste vide ([]), cela provoquera une division par zéro, ce qui générera une erreur**.**

```
def mystere(l):
    z = 0
    t = 0
```

```
    for i in range(len(l)):
        z += l[i]
        t += 1
    return z / t

print(mystere([1, 2, 3]))
```

```
2.0
```

5- ADN. Un brin d'ADN peut être modélisé par une chaîne de caractères ne contenant que des symboles "a", "t", "g", "c".

```
def est_brin_ADN(chaine):
    valid_chars = {'a', 't', 'g', 'c'}
    for char in chaine:
        if char not in valid_chars:
            return False
    return True

print(est_brin_ADN("atcgdadt"))
print(est_brin_ADN("atcgcadt"))

def association(nucleotide):
    complements = {'a': 't', 't': 'a', 'c': 'g', 'g': 'c'}
    return complements.get(nucleotide)

print(association('a'))
print(association('c'))

def replique(chaine_ADN):
    compl = ''
    for nucleotide in chaine_ADN:
        compl += association(nucleotide)
    return compl

print(replique("atcg"))
```

```
False
False
t
g
tagc
```

Exercice 16 – Dictionnaires

```
dict_eleve = {
    'nom': 'Figny',
```

```
    'prénom': 'Jean',
    'age': 16,
}

# Accéder au nom et au prénom de l'élève
nom_eleve = dict_eleve['nom']
prenom_eleve = dict_eleve['prénom']
print("Nom de l'élève:", nom_eleve)  # 'Figny'
print("Prénom de l'élève:", prenom_eleve)  # 'Jean'

# Nombre d'éléments dans le dictionnaire
nb_elements = len(dict_eleve)
print("Nombre d'éléments dans le dictionnaire:", nb_elements)  # 3

# Ajouter la moyenne de Jean
dict_eleve['moyenne'] = 12.34
print("Dictionnaire après ajout de la moyenne:", dict_eleve)

# Modifier l'âge de Jean
dict_eleve['age'] = 17
print("Dictionnaire après modification de l'âge:", dict_eleve)

# Supprimer la moyenne de Jean
dict_eleve.pop('moyenne', None)  # None est la valeur par défaut si la
clé n'existe pas
print("Dictionnaire après suppression de la moyenne:", dict_eleve)
```

```
Nom de l'élève: Figny
Prénom de l'élève: Jean
Nombre d'éléments dans le dictionnaire: 3
Dictionnaire après ajout de la moyenne: {'nom': 'Figny', 'prénom': 'Jean', 'age': 16, 'moyenne': 12.34}
Dictionnaire après modification de l'âge: {'nom': 'Figny', 'prénom': 'Jean', 'age': 17, 'moyenne': 12.34}
Dictionnaire après suppression de la moyenne: {'nom': 'Figny', 'prénom': 'Jean', 'age': 17}
```

Exercice 17 - Depuis un dictionnaire vide.

Cet exercice porte sur la manipulation des dictionnaires en Python. Il comporte plusieurs parties :

- Création d'un dictionnaire vide et vérification du type de l'objet.
- Création d'un dictionnaire utilisateur avec des clés et des valeurs spécifiées.
- Écriture d'une fonction pour convertir une liste de couples clés-valeurs en un dictionnaire.
- Écriture d'une fonction pour accéder à une valeur dans un dictionnaire en spécifiant une clé, avec une valeur par défaut si la clé n'existe pas.
- Écriture d'une fonction pour copier un dictionnaire en excluant une clé spécifique.

Chaque étape de l'exercice met en pratique différentes fonctionnalités et opérations courantes sur les dictionnaires en Python.

```
# 1. Créer un dictionnaire vide
dictionnaire_vide1 = {}
dictionnaire_vide2 = dict()

# 2. Vérifier si un objet est un dictionnaire
# Méthode 1 :
d = {}
print(isinstance(d, dict))  # True

# Méthode 2 :
print(type(d) == dict)  # True

# 3. Créer le dictionnaire utilisateur
utilisateur = {
    'nom': 'Josephe',
    'prenom': 'Apolline',
    'age': 22,
    'password': 'juygvfesw'
}

# 4. Écrire une fonction pour convertir une liste de couples clés-valeurs en un dictionnaire
def liste_couples_vers_dictionnaire(liste):
    dictionnaire = {}
    for couple in liste:
        cle, valeur = couple
        dictionnaire[cle] = valeur
    return dictionnaire

# Test de la fonction
entree = [('nom', 'Josephe'), ('prenom', 'Apolline'), ('age', 22), ('password', 'juygvfesw')]
sortie = liste_couples_vers_dictionnaire(entree)
print(sortie)

# 5. Écrire une fonction acceder
def acceder(d, k, defaut):
    if k in d:
        return d[k]
    else:
        return defaut

# Test de la fonction
print(acceder({'a': 1, 'b': 5}, 'a', 2))  # 1
print(acceder({'a': 1, 'b': 5}, 'c', 2))  # 2
```

```python
# 6. Écrire une fonction copier_sauf
def copier_sauf(d, k):
    copie = d.copy()
    copie.pop(k, None)
    return copie

# Test de la fonction
print(copier_sauf({"a": 1, "b": 2, "c": 3}, "a"))  # {"b": 2, "c": 3}
```

Exercice 18 - Itérer sur un dictionnaire.

L'objectif de cet exercice est de pratiquer la manipulation de dictionnaires en Python ainsi que les opérations courantes associées à ces structures de données.

```python
# 1. Itérer sur un dictionnaire
scores_pacman = {'paul': 34, 'honorine': 456, 'marcel': 89, 'octave':
542, 'marine': 12, 'mélanie': 134, 'patricia': 631}

# 2. Ajouter le score d'Amandine
scores_pacman['amandine'] = 542

# 3. Fonction pour obtenir le score d'un joueur
def obtenir_score(dictionnaire, joueur):
    return dictionnaire.get(joueur.lower(), 0)

# 4. Fonction pour inscrire un joueur
def incrire_joueur(score_jeu, joueur):
    joueur = joueur.lower()
    if joueur not in score_jeu:
        score_jeu[joueur] = 0
        return True
    else:
        return False

# 5. Fonction pour obtenir la liste des noms de joueurs
def liste_joueurs(dictionnaire):
    return list(dictionnaire.keys())

# 6a. Calcul du score moyen avec sum
score_moyen = sum(scores_pacman.values()) / len(scores_pacman)

# 6b. Calcul du score moyen sans sum
somme_scores = 0
for score in scores_pacman.values():
    somme_scores += score
score_moyen_sans_sum = somme_scores / len(scores_pacman)
```

```python
# 7. Fonction pour générer une chaîne de caractères contenant les
scores des joueurs
def scores_des_joueurs(dictionnaire):
    texte = ""
    for joueur, score in dictionnaire.items():
        texte += f"Le score de {joueur.capitalize()} est {score}\n"
    return texte

# 8. Fonction pour retourner une liste de phrases
def scores_des_joueurs_liste(dictionnaire):
    liste_phrases = []
    for joueur, score in dictionnaire.items():
        phrase = f"Le score de {joueur.capitalize()} est {score}"
        liste_phrases.append(phrase)
    return liste_phrases

# 9a. Fonction pour retourner la liste des phrases triées par ordre
alphabétique des prénoms
def scores_des_joueurs_tries_alpha(dictionnaire):
    liste_phrases_triees = []
    for joueur, score in sorted(dictionnaire.items()):
        phrase = f"Le score de {joueur.capitalize()} est {score}"
        liste_phrases_triees.append(phrase)
    return liste_phrases_triees

# 9b. Fonction pour retourner la liste des phrases triées par score
croissant
def scores_des_joueurs_tries_score(dictionnaire):
    liste_phrases_triees = []
    for joueur, score in sorted(dictionnaire.items(), key=lambda x:
x[1]):
        phrase = f"Le score de {joueur.capitalize()} est {score}"
        liste_phrases_triees.append(phrase)
    return liste_phrases_triees

# Tests des différentes fonctions
print(obtenir_score(scores_pacman, 'Amandine'))  # 542
print(incrire_joueur(scores_pacman, 'Amandine'))  # False
print(liste_joueurs(scores_pacman))
print(score_moyen)
print(score_moyen_sans_sum)
print(scores_des_joueurs(scores_pacman))
print(scores_des_joueurs_liste(scores_pacman))
print(scores_des_joueurs_tries_alpha(scores_pacman))
print(scores_des_joueurs_tries_score(scores_pacman))
```

```
542
False
['paul', 'honorine', 'marcel', 'octave', 'marine', 'mélanie', 'patricia', 'amandine']
305.0
305.0
Le score de Paul est 34
Le score de Honorine est 456
Le score de Marcel est 89
Le score de Octave est 542
Le score de Marine est 12
Le score de Mélanie est 134
Le score de Patricia est 631
Le score de Amandine est 542

['Le score de Paul est 34', 'Le score de Honorine est 456', 'Le score de Marcel est 89', 'Le score de Octave est 542', 'Le score de Marine est 12', 'Le score de Mélanie est 134', 'Le score de
['Le score de Amandine est 542', 'Le score de Honorine est 456', 'Le score de Marcel est 89', 'Le score de Marine est 12', 'Le score de Mélanie est 134', 'Le score de Octave est 542', 'Le scor
['Le score de Marine est 12', 'Le score de Paul est 34', 'Le score de Marcel est 89', 'Le score de Mélanie est 134', 'Le score de Honorine est 456', 'Le score de Octave est 542', 'Le score de
```

Exercice 19 - Complément sur les boucles while

```python
import random

# 1. Boucle while équivalente à la boucle for donnée
mot = "raisonable"
c = 0
i = 0
while i < len(mot):
    if mot[i] == 'a':
        c = c + 1
    i += 1

# 2. Boucle while pour doubler le capital
capital = 1000
annees = 0
while capital < 2000:
    capital *= 1.05
    annees += 1

# 3. Fonction pour jouer au jeu du "plus ou moins"
def plus_ou_moins():
    nombre_aleatoire = random.randint(1, 100)
    essais = 0
    while True:
        essai = int(input("Votre nombre : "))
        essais += 1
        if essai < nombre_aleatoire:
            print("C'est plus")
        elif essai > nombre_aleatoire:
            print("C'est moins")
        else:
```

```
            print(f"BRAVO ! Le nombre était {nombre_aleatoire}, vous
avez trouvé en {essais} coups.")
            break

# 4. Fonction pour choisir un entier aléatoire entre 1 et 10 non
présent dans la liste
def entier_aleatoire_un_dix(liste):
    disponibles = [i for i in range(1, 11) if i not in liste]
    if disponibles:
        return random.choice(disponibles)
    else:
        return -1

# 5. Fonction pour générer un échantillon avec remise
def echantillon(a, b, n):
    return [random.randint(a, b) for _ in range(n)]

# 6. Fonction pour générer un échantillon sans remise
def echantillon_sans_remise(a, b, n):
    assert b - a + 1 >= n, "Pas assez de valeurs entre a et b"
    valeurs_disponibles = list(range(a, b + 1))
    return random.sample(valeurs_disponibles, n)

# 7. Fonction pour la conjecture de Syracuse
def syracuse_vol(n):
    vol = [n]
    while n != 1:
        if n % 2 == 0:
            n = n // 2
        else:
            n = 3 * n + 1
        vol.append(n)
    return vol

def syracuse_duree(n):
    vol = syracuse_vol(n)
    return len(vol)

def syracuse_hauteur(n):
    vol = syracuse_vol(n)
    return max(vol)
```

Conclusion :

Ces travaux pratiques sur Python dans Google Colab offrent une immersion riche dans le langage de programmation Python, en mettant l'accent sur des concepts fondamentaux et avancés. Les exercices couvrent divers aspects, allant de la manipulation des types de données de base tels que les entiers, les flottants et les booléens, à des sujets plus avancés tels que les boucles, les fonctions, les dictionnaires et les listes.

- L'utilisation de Google Colab comme environnement de développement est pertinente car il offre un accès facile à un environnement de notebook interactif basé sur le cloud, ce qui facilite la collaboration et l'exécution de code Python sans nécessiter une configuration complexe.

- L'exploration de Python dans le contexte de l'analyse de données et de la prise de décision est particulièrement pertinente. La maîtrise de Python s'avère cruciale pour les professionnels et les chercheurs qui souhaitent analyser,