Nom :	Prénom :	Classe :	Note :

I Autour des listes et des chaînes

Q 1/3 : *Considérons la chaîne de caractères chaine1='1234' ainsi que les trois listes suivantes liste0=['abcd','ef',1,2,3] liste1=[1,2,4,8] et liste2=[1,2,[3,4,[5,6]],[1,2,3]]. Les questions font référence à ces données.*
Précisez ce que vous retourne l'interpréteur python pour les commandes en lignes suivantes :

len(liste2)	4	liste2[2]=6	[1, 2, 6, [1, 2, 3]]
chaine1[2]	'3'	len(liste2[2])	3
liste0[1][1]	'f'	liste2[2][0]==3	True
liste0[1][2]	IndexError: string index out of range	liste0[1][0]=='e'	True
liste2[1]	2	liste1+ liste2	[1, 2, 4, 8, 1, 2, [3, 4, [5, 6]], [1, 2, 3]]
len(chaine1)	4	chaine1+ liste0	TypeError: Can't convert 'list' object to str implicitly
2* liste2	[1, 2, [3, 4, [5, 6]], [1, 2, 3], 1, 2, [3, 4, [5, 6]], [1, 2, 3]]	liste2[2][2][0]==5	True

Q 2/2 : Écrire une fonction python nommée ***smul*(a, liste)** à deux paramètres, un nombre ***a*** et une liste de nombres ***liste***, qui multiplie chaque élément de la liste ***liste*** par le nombre ***a*** et renvoie une nouvelle liste : ***smul*(2, [1, 2, 3]) → [2, 4, 6].**

```
def smul(a, liste):
    liste_mul = []
    for i in range(len(liste)):
        liste_mul.append(a*liste[i])
    return liste_mul
```

Q 3/2 : Ecrire une fonction python ***vsom*(liste1, liste2)** qui prend en paramètre deux listes de nombres de même longueur et qui renvoie une nouvelle liste constituée de la somme terme à terme de ces deux listes :
***vsom*([1, 2, 3], [4, 5, 6]) → [5, 7, 9].**

```
def vsom(liste1, liste2):
    out = []
    if len(liste1) != len(liste2):
        # On renvoie la liste vide si les deux liste n'ont pas le même
        # nombre d'éléments (on pourrait aussi renvoyer le booléen False)
        return []
    for i in range(len(liste1)):
        out.append(liste1[i] + liste2[i])
    return out
```

Q 4/2 : Écrire une fonction python ***vdif*(liste1, liste2)** qui prend en paramètre deux listes de nombres de même longueur et qui renvoie une nouvelle liste constituée de la différence terme à terme de ces deux listes (la première moins la deuxième) :
***vdif*([1, 2, 3], [4, 5, 6]) → [-3, -3, -3].**
Vous ferez uniquement appel aux fonctions ***smul*** et ***vsom*** développées aux deux précédentes questions.

```
27 def vdif(liste1 , liste2):
28     return vsom(liste1,smul(-1,liste2))
```

Q 5/2 : *Parmi les affirmations suivantes, indiquez celle ou celles qui sont vraies.*

```
4 def mystere(L):
5     S = 0
6     for i in range(len(L)-1):
7         S=S+L[i]
8     return S
```

- ☐ somme([1, 2, 3, 4]) renvoie 10.
- ■ somme([1, 2, 3, 4]) renvoie 6.
- ☐ somme([1, 2, 3, 4]) renvoie 9.
- ☐ somme([1, 2, 3, 4]) renvoie un message d'erreur.

Q 6/3 : *La fonction Python suivante a deux arguments d'entrée N et L qui sont des chaînes de caractères.*
Parmi les affirmations suivantes, cochez celles ou celles qui sont vraies.

```
8  def test(L,N):
9      if len(L)<=len(N):
10         bool=True
11         for i in range(len(L)):
12             if L[i]!=N[i]:
13                 bool=False
14         return bool
15     else:
16         return False
```

- ☐ test(L,N) renvoie True si et seulement si les deux chaînes de caractères L et N sont distinctes.
- ☐ test(L,N) renvoie True si et seulement si les deux chaînes de caractères L et N sont égales.
- ■ test('CONTROLE','CONTROLEUR') renvoie True.
- ☐ test('RIEN,'AERIEN') renvoie un message d'erreur.

Q 7/3 : On considère la fonction Python suivante (on suppose que l'on a accès fonction test de la question précédente Q6).
Rappel : à la rencontre du premier « return », on sort de la fonction **cherche(.)**

```
18 def cherche(mot, chaine):
19     i=0
20     while i<=len(chaine)-len(mot):
21         if chaine[i]==mot[0] and test(mot, chaine[i:]):
22             return i
23         i=i+1
24     return -1
```

Parmi les affirmations suivantes, indiquez celle ou celles qui sont vraies.

- ☐ La fonction ***cherche*** renvoie le nombre de fois où un mot apparaît dans une chaîne de caractères.
- ■ La fonction ***cherche*** renvoie l'index de la première lettre de l'apparition de ***mot*** dans ***chaîne***.
- ☐ La fonction ***cherche*** renvoie les indexes de la première lettre de chaque apparition de ***mot*** dans ***chaîne***.
- ■ La fonction ***cherche*** renvoie -1 si la recherche échoue.
- ☐ La fonction ***cherche*** renvoie -1 systématiquement.
- ☐ La fonction ***cherche*** ne renvoie rien.

La décimation est un traitement qui consiste à ne prélever dans une liste qu'un élément sur k. Cet opérateur est commun en traitement du signal. Il a pour conséquence un étalement du spectre de la suite de nombre dans un rapport k.

Q 8/3 : Écrire la fonction $y = decimation(s, k)$ qui a pour arguments d'entrée la liste s définie précédemment et le rapport de décimation k. La liste retournée y est la décimation de s dans un rapport k.

```
def decimation(s, k):
    return s[::k]  # Les valeurs de la première à la dernière de k en k

def decimation(s, k):
    # Alternative plus lente
    y = []
    i = 0
    while i < len(L):
        y.append(s[i])
        i += k
    return y
```

II Revisite d'un sujet abordé lors des TPs

On considère la somme double définie par : $s = \sum_{1 \le i \le j \le n} \frac{i}{j}$

Dans cette expression, n est une donnée du problème, un paramètre d'entrée. Le calcul de la somme s est intégré à une fonction $triangulaire(n)$ qui retourne le résultat s.

Q 9/2 : Proposer le pseudocode du calcul de la somme définie par s.

```
# calcul de la somme s = ∑1≤i≤j≤n i/j
# n passé en paramètre dans la fonction

s ← 0                          # initialisation de la variable s à 0
pour j de 1 à n faire :
    pour i de 1 à j+1 faire :
        s ← s+i/j              #  mise à jour de la variable s, calcul de la somme
```

Q 10/3 : Ecrire la fonction $triangularisation(n)$ prenant en argument un entier n et renvoyant la valeur de s. Cette fonction intègre le traitement proposé à la question précédente.

```
12 def triangularisation(n):
13     s = 0
14     for j in range(1,n+1):
15         for i in range(1,j+1):
16             s+= i/j
17     return(s)
```

Q 11/3 : Commenter les lignes de la fonction « ma_fonction » définie pas le script suivant.

```
23 def ma_fonction(m):
24     E=[]
25     a,b=100,2000
26     for j in range(1,m+1):
27         n=randint(a,b)
28         r0=n*(n+3)/4
29         r=triangularisation(n)
30         E.append(r-r0)
31     return(E)
```

Votre analyse de cet algorithme

Fonction de nom « ma_foction », l'argument passé en paramètre est un entier m
Initialisation : E est une liste vide
Affectation multiple : a=100 et b=2000
Boucle for d'index j=1,…,m
Génération des réalisations : n est un entier aléatoire de support [a,…,b].
r0 est le résultat attendu de la somme
Le résultat retourné par la fonction « triangularisation » est affecté à r
r-r0 est l'écart qui est concaténé à l liste E
La fonction retourne la liste des écarts de m réalisations.
Une aplication classique est l'estimation de la loi des écarts, moyenne et variance.

Concernant la fonction randint de la bibliothèque random

```
>>> import random
>>> help(random.randint)
Help on method randint in module random:

randint(self, a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
```

La fonction précédente retour la liste suivante pour $m = 100$ réalisations :

```
>>> E=ma_fonction(100)
>>> E
[1.6007106751203537e-10, 0.0, 0.0, 4.656612873077393e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 0.0, 3.2014213502407074e-10, 4.656612873077393e-10, 4.656612873077393e-10, 4.656612873077393e-10, 1.6007106751203537e-10, 0.0, 4.656612873077393e-10, 4.656612873077393e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 1.6007106751203537e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 0.0, 0.0, 0.0, 3.2014213502407074e-10, 4.656612873077393e-10, 0.0, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 0.0, 0.0, 4.656612873077393e-10, 0.0, 4.656612873077393e-10, 4.656612873077393e-10, 4.656612873077393e-10, 4.656612873077393e-10, 0.0, 4.656612873077393e-10, 3.2014213502407074e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 4.656612873077393e-10, 0.0, 4.5474735508864641e-13, 0.0, 4.656612873077393e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 0.0, 4.656612873077393e-10, 4.656612873077393e-10, 0.0, 4.656612873077393e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 1.6007106751203537e-10, 0.0, 3.2014213502407074e-10, 4.656612873077393e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 0.0, 4.656612873077393e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 3.2014213502407074e-10, 1.6007106751203537e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10, 4.656612873077393e-10, 3.2014213502407074e-10, 4.656612873077393e-10]
>>>
```

Q 12/2 : Que pensez-vous de ces résultats par rapport à ceux attendus ?

Pêle-mêle, nous pouvons admettre comme explication sur l'origine de ces écarts :

1- Les problèmes de troncatures liées au codage des nombres qui sont à l'origine d'erreurs cumulées,
2- Les problèmes d'absorptions : la somme de deux nombres qui sont dans un rapport important. Le plus petit s'efface par rapport au second.

Ils peuvent illustrer aussi ces problèmes d'absorption et de troncature par la somme harmonique :

$$\sum_{j=1,\dots,n} \frac{1}{j} \neq \sum_{j=n,\dots,1} \frac{1}{j}$$