Structures de données	Chap 01	Programmation orientée objet

Prénom :

Nom :

Exercice 1 — Vocabulaire — 5 pts

On considère le code suivant :

```
class Animal:
    def __init__(self, espece,age):
        self.espece = espece
        self.age=age
    def crier(self):
        print("Grrr !")
Gromit = Animal("chien")
```

1.Expliquer ce que sont les entités suivantes :

- `__init__`
- `espece`
- `crier`
- `Gromit`
- `Animal`

On considère le même code un peu modifié :

```
class Animal:
    def __init__(self, e,age):
        self.espece = e
        self.age = age
    def crier(self):
        print("Grrr !")
Gromit = Animal("chien")
Gromit.crier()
```

2. Est-ce le changement dans `__init__` est un problème, justifier.
3. Que se passe-t-il si on exécute le code précédent ?
4. Ecrire une méthode qui permet de déterminer si un animal a plus de 10 ans.

Exercice 2 — Lecture de code — 4 pts

On considère le code suivant :

```
1  class Compteur:
2      def __init__(self, depart=0):
3          self.valeur = depart
4      def inc(self,pas):
5          self.valeur = self.valeur + pas
6      def reset(self):
7          self.valeur = 0
8
9  c1 = Compteur(5)
10 c2 = Compteur()
11 c1.inc(1)
12 c2.inc(3)
13 c2.reset()
14 c2.inc(2)
15 print(c1.valeur)
16 print(c2.valeur)
```

1. Que va afficher la ligne 14, pourquoi ?
2. Que va afficher la ligne 15, pourquoi ?

Exercice 3 — Je sais que cela vous manquait – 2 pts

1. Ecrire une fonction récursive `maximum(liste)` qui, étant donnée une liste d'entier renvoie le maximum de cette liste.

Il faut utiliser le fait que le maximum d'une liste, c'est le maximum entre le premier element et le maximum du reste de la liste.

Exercice 4 — Surcharge de l'opérateur + — 2 pts

En Python, l'opérateur + peut être redéfini pour nos propres classes grâce à la méthode spéciale `__add__`.
On veut représenter une note scolaire avec sa valeur et son coefficient.

```
class Note:
    def __init__(self, valeur, coeff=1):
        self.valeur = valeur
        self.coeff = coeff
    def __str__(self):
        return "str(self.valeur)" + "coeff" + str(self.coeff)"
    def __add__(self,note):
        new_valeur =
        new_coeff =
        return ...
```

1. Compléter la classe pour que l'opération + entre deux notes renvoie une **nouvelle Note** dont :
   - la valeur est la **moyenne pondérée** des deux notes,
   - le coefficient est la somme des coefficients.
2. Que doit afficher le programme suivant ?

```
n1 = Note(10, 3)
n2 = Note(18, 1)
n3 = n1 + n2
print(n3.valeur, n3.coeff)
```