# Séance 9: Initiation aux objets*

L1 – Université Nice Sophia Antipolis

**Exercice 1 (Cercles, ⋆)**

1. *Dites précisément ce qu'on entend par attribut d'instance, méthode d'instance, attribut de classe. Qu'appelle-t-on l'état d'un objet ?*
2. *Dans le code ci-dessous, pourrait-on faire de* `cercles` *une variable initialisée à l'extérieur de la classe Cercle ?*

```
class Cercle :
    """La classe des Cercles en L1"""
    cercles = []              # une variable de classe

    def __init__(self,centre,rayon=1,couleur='red') :
        self.centre = centre
        self.rayon = rayon
        self.couleur = couleur
        Cercle.cercles.append(self)
```

□

**Exercice 2 (Vecteurs, ⋆⋆)**
*Programmons une classe* `Vect2d` *dont les objets modéliseront des vecteurs à deux composantes réelles. Lorsque le mathématicien dit : « Soit le vecteur », le programmeur Python dira :* `u = Vect2d(3,-2)`.

1. *Programmez le constructeur, chargé d'initialiser les champs* `x` *et* `y` *de l'instance courante, qui se nomme* `self`.

   *Par défaut,* `x` *et* `y` *seront mis à 0.*
2. *Programmez une méthode chargée de produire une chaîne de caractères représentant l'état de l'objet. Cette fonction porte en Python un nom particulier !*
3. *Si je veux additionner deux vecteurs, j'ai deux solutions. Ou bien je programme une fonction mathématique* `add(v1,v2)` *à l'extérieur de la classe. Ou bien – dans une optique de pure programmation par objets – je programme* `add` *comme méthode d'instance à l'intérieur de la classe. Adoptez cette seconde solution, pour laquelle* `add` *devient un message envoyé à un vecteur. Le résultat de l'addition sera un nouveau vecteur, aucune mutation !*
4. *Programmez une méthode* `mul_ext` *réalisant la multiplication d'un réel* $k$ *par un vecteur. Le résultat sera un nouveau vecteur, aucune mutation.*
5. *Programmez une méthode zoom demandant à un vecteur de bien vouloir se multiplier par k et d'être ainsi définitivement modifié ! Faites bien la différence avec* `mul_ext`.
6. *Programmez une méthode* `prodscal` *demandant à un vecteur de retourner son produit scalaire avec un autre vecteur.*
7. *En déduire une méthode* `norme` *demandant à un vecteur de retourner sa longueur.*

*Remerciements: ce TD a été conçu par Jean-Paul Roy.

8. *Programmez à l'extérieur de la classe une fonction* `add(v1,v2)` *retournant la somme vectorielle .*
   *N.B. Est-il possible d'écrire $u + v$ si $u$ et $v$ sont deux vecteurs ? En principe non, mais en Python, des méthodes spéciales sont cachées sous les opérateurs. Si votre méthode add se nomme* `__add__`*, cela devient possible !*

□

**Exercice 3 (Degrés, $\star\star$)**
*On souhaite modéliser un climatiseur par la classe* `Climatiseur`*. Un climatiseur permet d'obtenir une température souhaitée en degrés Celsius. Lorsque je commande un climatiseur à l'usine, je spécifie la plage de températures de fonctionnement $[tmin, tmax]$ : il ne descendra jamais en-dessous de $tmin$ et ne dépassera jamais $tmax$. Je peux aussi spécifier la température souhaitée au démarrage $tinit$. Si je ne la demande pas, l'usine le règlera à $20°C$ par défaut. Un climatiseur est muni de deux commandes pour augmenter ou diminuer la température souhaitée de $1°C$ (en restant dans la plage de fonctionnement). Il a enfin un bouton afficher qui affiche avec print la température à la fois en $°C$ et en $^{circ}F$. On rappelle que $°C = (°F - 32) * \frac{9}{5}$.* □