# La Programmation Orientee Objet (Python)

Objectifs maitriser les notions suivante

Notion de classe : Constructeur

Notion d'objet : Instanciation

Notion : Héritage

**Exercice 1 :**

Définir une classe MaClasse possédant les attributs suivants :

données : deux attributs de classes : x = 23 et y = x + 5.méthode : une méthode affiche contenant un attribut d'instance z = 42 et les affichages de y et de z.

Dans le programme principal, instanciez un objet de la classe MaClasse et invoquez la méthode affiche.

2. Définir une classe Vecteur2D avec un constructeur fournissant les coordonnées par défaut d'un vecteur du plan (par exemple : x = 0 et y = 0).

Dans le programme principal, instanciez un Vecteur2D sans paramètre, un Vecteur2D avec ses deux paramètres, et affichez-les.

3. Enrichissez la classe Vecteur2D précédente en lui ajoutant une méthode d'affichage et une méthode de surcharge d'addition de deux vecteurs du plan.

Dans le programme principal, instanciez deux Vecteur2D, affichez-les et affichez leur somme.

**Exercice 2:**

1. Définir une classe Rectangle avec un constructeur donnant des valeurs (longueur et largeur) par défaut et un attribut nom = "rectangle", une méthode d'affichage et une méthode surface renvoyant la surface d'une instance.

Définir une classe Carre héritant de Rectangle et qui surcharge l'attribut d'instance : nom = "carré".

Dans le programme principal, instanciez un Rectangle et un Carre et affichez-les

2. Définir une classe Point avec un constructeur fournissant les coordonnées par défaut d'un point du plan (par exemple : x = 0.0 et y = 0.0).

Définir une classe Segment dont le constructeur possède quatre paramètres : deux pour l'origine et deux pour l'extrémité. Ce constructeur définit deux attributs : orig et extrem,

instances de la classe Point. De cette manière, vous concevez une classe composite : La classe Segment est composée de deux instances de la classe Point

Ajouter une méthode d'affichage.

Enfin écrire un auto-test qui affiche une instance de Segment initialisée par les valeurs1, 2, 3 et 4.

**Exercice 3:**

1. Créer une **classe Calcul** ayant un **constructeur par défaut** (sans paramètres) permettant d'effectuer différents calculs sur les nombres entiers.
2. Créer au sein de la classe Calcul une **méthode** nommée **Factorielle()** qui permet de calculer le factorielle d'un entier. Tester la méthode en faisant une instanciation sur la classe.
3. Créer au sein de la classe Calcul une **méthode** nommée **Somme()** permettant de calculer **la somme des n premiers entiers: 1 + 2 + 3 + .. + n.** Tester la méthode.
4. Créer au sein de la classe Calcul une **méthode** nommée **testPrim()** permettant de tester **la primalité d'un entier** donné. Tester la méthode.
5. Créer au sein de la classe Calcul une **méthode** nommée **testPrims()** permettant de tester **si deux nombres sont premier entre eux.**
6. Créer une **méthode tableMult()** qui crée et affiche la table de multiplication d'un entier donné. Créer ensuite une méthode **allTablesMult()** permettant d'afficher toutes les tables de multiplications des entiers 1, 2, 3, ..., 9.
7. Créer une **méthode listDiv()** qui récupère tous **les diviseurs d'un entier** donné sur **une liste Ldiv**. Créer une autre méthode **listDivPrim()** qui récupère tous les diviseurs **premiers** d'un entier donné