Exercice 1.
Définir une classe `Point` pour représenter un point du plan. Cette classe possède deux attributs, `_x` et `_y` qui sont des réels et désignent respectivement l'abscisse et l'ordonée du point.

1. Ecrire le constructeur de cette classe et une méthode `coordonnees(self)` qui retourne le couple `(_x,_y)`.
2. Ajouter une méthode `norme(self)` qui retourne la distance entre l'origine du repère et le point.
3. Ajouter une méthode `distance(self, p)` qui retourne la distance entre `self` et le point p.
4. Ajouter une méthode `__str__(self)` qui renvoie une chaîne de caractère de la forme `(4;3)`.
5. Tester ces opérations.

Exercice 2.
Définir une classe `Cercle` qui représente un cercle du plan. Cette classe possède deux attributs, `_centre` qui représente le centre du cercle et qui est de type `Point` et son rayon `_r`.

1. Ecrire le constructeur de cette classe.
2. Ecrire une méthode `perimetre(self)` et une méthode `surface(self)` qui retourne respectivement le périmètre et la surface du cercle.
3. Ecrire une méthode `__str__(self)` qui retourne une chaîne de caractère de la forme `(4;3) 2`.
4. Ecrire une méthode `appartient(self, p)` qui retourne `True` si le point `p` appartient au cercle et `False` sinon.
5. Tester ces opérations.

Exercice 3.
Définir une classe `Date` pour représenter une date, avec trois attributs `jour`, `mois`, `annee`.

1. Ecrire son constructeur.
2. Ajouter une méthode `__str__` qui renvoie une chaîne de caractères de la forme `"8 mai 1945"`. On pourra se servir d'un attribut de classe donnant les noms des douzes mois de l'année. Tester en construisant des objets de la classe `Date` puis en les affichant avec **`print`**.
3. Ajouter une méthode `__lt__` qui permet déterminer si une date `d1` est antérieur à une date `d2` en écrivant `d1 < d2`. La tester.

Exercice 4.
Dans certains langages de programmation, comme Pascal ou Ada, les tableaux ne sont pas forcément indexés à partir de 0. C'est le programmeur qui choisit sa plage d'indices. Par exemple, on peut déclarer un tableau dont les indices vont de -10 à 9 si on le souhaite. Dans cet exercice, on se propose de construire une classe `Tableau` pour réaliser de tels tableaux.
Un objet de cette classe aura deux attributs, un attribut `premier` qui est la valeur de premier indice et un attribut `contenu` qui est un tableau Python contenant les éléments. Ce dernier est un vrai tableau Python, indexé à partir de 0.

1. Ecrire un constructeur `__init__(self, imin, imax, v)` où `imin` est le premier indice, `imax` est le dernier indice et `v` la valeurutilisée pour initialiser toutes les cases du tableau. Ainsi, on peut écrire `t = Tableau(-10,9,42)` pour construire un tableau de vingt cases, indexées de -10 à 9 et toutes initialisées avec la valeur 42.
2. Ecrire une méthode `__len__(self)` qui renvoie la taille du tableau.
3. Ecrire une méthode `__getitem__(self, i)` qui renvoie l'élément du tableau `self` d'indice `i`. De même écrire une méthode `__setitem__(self, i, v)` qui modifie l'élément d'indice `i` pour lui donner la valeur `v`. Ces deux méthodes doivent vérifier que l'indice `i` est bien valide et , dans le cas contraire, lever l'exception `IndexError` avec la valeur `i` en argument (**`raise`** `IndexError(i)`).
4. Enfin, écrire une méthode `__str__(self)` qui renvoie une chaîne de caractères décrivant le contenu du tableau.

Exercice 5.
On veut définir une classe `TaBiDir` pour des tableaux bidirecttionnels, dont une partie des éléments ont des indices positifs et une partie des éléments des indices négatifs, et qui sont extensibles aussi bien par la gauche que la droite. Plus précisément, les indices d'un tel tableau bidirectionnel vont aller d'un indice i_{min} à un indice i_{max}, tous deux inclus, et tels que $i_{min} \leq 0$ et $-1 \leq i_{max}$. Le tableau bidirectionnel vide correspond au cas où i_{min} et i_{max} vat -1.
La classe `TaBiDir` a pour attributs deux tableaux Python : un tableau `droite` contenant l'élément d'indice `0` et les autres éléments d'indices positifs, et un tableau `gauche` tel que `gauche[0]` contient l'élément d'indice `-1` du tableau bidirectionnel, et `gauche[1]` et `gauche[2]`, etc. contiennent les éléments d'indices négatifs suivants, en progressant vers la gauche.

- Ecrire un constructeur `__init__(self, g, d)` construisant un tableau bidirectionnel contenant, dans l'ordre, les éléments des tableaux `g` et `d`. Le dernier élément de `g` (si `g` n'est pas vide), devra être calé sur l'indice `-1` du tableau bidirectionnel, et le premier élément de `d` (si `d` n'est pas vide) sur l'indice `0`. Ecrire également les méthodes `imin(self)` et `imax(self)` renvoyant respectivement l'indice minimum et maximum.
- Ajouter une méthode `append(self, v)`, qui ajoute un élément `v` à droite du tableau bidirectionnel et une méthode `prepend(self, v)` ajoutant l'élément `v` à gauche du tableau bidirectionnel.
- Ajouter une méthode `__getitem__(self, i)` qui renvoir l'élément du tableau bidirectionnel `self` à l'indice `i`, et une méthode `__setiem__(self, i)` qui modifie l'élément du tableau `self` d'indice `i` pour lui donner la valeur `v`.
- Ajouter une méthode `__str__(self)` qui renvoie une chaîne de caractères décrivant le contenu du tableau.