Exercice 1. — Point du plan

Écrire une classe nommée `Point2D` admettant deux attributs (*syn.* variables d'instances en Python) l'abscisse et l'ordonnée d'un point du plan ainsi que les méthodes d'instance suivantes :

- `affiche` qui affiche les coordonnées `x` et `y` de l'objet instancié (i.e. un point) sous la forme `(x ; y)` ;
- `distance_a_0` qui renvoie la distance à l'origine du point ;
- `distance` qui reçoit en paramètre une autre instance de la classe `Point2D` (i.e. un point) et renvoie la distance entre ces deux points ;
- `est_confondu` qui reçoit en paramètre un autre point et renvoie `True` si les deux points sont confondus, `False` sinon.

Exercice 2. — Carré

Écrire une classe `Carre` admettant un attribut, la longueur d'un côté du carré, et deux méthodes d'instance :

- `perimetre` qui renvoie le périmètre du carré ;
- `aire` qui renvoie l'aire du carré.

Exercice 3. — Triangle

Définir une classe `Triangle` avec trois attributs, les longueurs respectives des trois côtés, et deux méthodes d'instance :

- `aire` qui renvoie l'aire du triangle à l'aide de la formule de Héron :
$$\mathcal{A} = \sqrt{p(p-a)(p-b)(p-c)}$$
où a, b, et c représentent les trois longueurs du triangle et p, le demi-périmètre.
- `est_rectangle` qui renvoie `True` si le triangle est rectangle, `False` sinon.

Exercice 4. — Temps

Écrire une classe `Temps` qui permet de définir un horaire au format « `hh:mm:ss` » et qui admet les méthodes suivantes :

- `affiche` qui affiche l'horaire au format « 12 h 37 min 45 s » ;
- `__add__` qui ajoute deux horaires de la classe `Temps` ;
- `__sub__` qui calcule la différence entre deux horaires de la classe `Temps`.

Exercice 5. — Date

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

- Écrire son constructeur.
- 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 qui est un tableau donnant les noms des douze mois de l'année. Tester en instanciant des objets de la classe `Date` puis en les affichant avec `print`.
- Ajouter une méthode `__lt__` qui permet de déterminer si une date `d1` est antérieure à une date `d2` en écrivant `d1 < d2`. La tester.

Exercice 6. — Fraction

1. Écrire une classe `Fraction` qui vérifie les instructions suivantes.

```
>>> x = Fraction(12, 9)
>>> x.num
12
>>> x.den
9
>>> x
(12 / 9)
>>> x.simplifier()
(4 / 3)
>>> y = Fraction(48, 36)
>>> x == y
True
>>> z = Fraction(8, 10)
>>> x < z
False
>>> x + z
(32 / 15)
>>> x * z
(16 / 15)
```

2. Modifier le « constructeur » de la classe `Fraction` de sorte qu'il lève l'exception
   - `TypeError` si le numérateur ou le dénominateur ne sont pas des entiers ;
   - `ValueError` si le dénominateur n'est pas un entier strictement positif.

   Faire une recherche sur l'instruction `raise`.

Exercice 7. — Écrire une classe `Vecteur` admettant trois attributs (ses coordonnéees dans l'espace) comportant :
- une méthode permettant d'afficher les coordonnées du vecteur sous la forme `(x ; y ; z)` ;
- une méthode renvoyant le vecteur somme de deux vecteurs (en surchargeant `__add__`) ;
- une méthode renvoyant la norme du vecteur ;
- une méthode renvoyant le produit scalaire de deux vecteurs ;
- une méthode renvoyant `True` si un vecteur est colinéaire à un autre (passé en paramètre), `False` sinon ;
- une méthode permettant de vérifier si un vecteur est orthogonal à un autre.