Exercice 1 :

1. Définir une classe *Livre* avec les attributs suivants : *titre, auteur (Nom complet), prix.*
2. Définir les méthodes d'accès aux différents attributs de la classe.
3. Définir un constructeur permettant d'initialiser les attributs de la méthode par des valeurs saisies par l'utilisateur.
4. Définir la méthode *afficher() -> str* permettant d'afficher les informations du livre en cours.
5. Écrire un programme testant la classe Livre.

Exercice 2 :

1. Définir une classe Rectangle ayant les attributs suivants : *longueur et largeur.*
2. Définir à l'aide des propriétés les méthodes d'accès aux attributs de la classe.
3. Ajouter les méthodes suivantes :
   — *perimetre() -> float* : retourne le périmètre du rectangle.
   — *aire() -> float* retourne l'aire du rectangle.
   — est_carre() -> bool : vérifie si le rectangle est un carré.

Exercice 3 :

1. Écrire une classe Complexe permettant de représenter des nombres complexes. Un nombre complexe Z comporte une partie réelle et une partie imaginaire : Z = *reel* + *imaginaire* * i
2. Définir les méthodes d'accès aux attributs de la classe.
3. Définir un constructeur d'initialisation pour la classe.
4. Ajouter la méthode *addition(Complexe) -> Complexe* : ajoute (sans le modifier) au nombre en cours un nombre complexe passé en argument et retourne le nombre complexe obtenu.
5. Ajouter la méthode *soustraction(Complexe) -> Complexe* : soustraie (sans le modifier) au nombre en cours un nombre complexe passé en argument et retourne le nombre complexe obtenu.
6. Ajouter la méthode *afficher() -> str* : Elle donne une représentation d'un nombre complexe comme suit : a+b*i.
7. Écrire un programme permettant de tester la classe Complexe.

Exercice 4 : En Python, la fonction *___init___* est nommée *constructeur*. Elle est appelée automatiquement lors de l'instanciation de l'objet. Il existe un certain nombre d'autres fonctions particulières, chacune avec un nom prédéfini et entourée d'une paire de _. Tous les opérateurs classiques de Python peuvent être ainsi redéfinis pour un objet :

— *___eq___* définit l'opérateur ==
— *___lt___* définit l'opérateur <
— *___contains___* définit l'opérateur *in*

1. Définir une classe *Loto*. Cette classe possède un attribut *numeros* de type *list* qui contiendra les 6 numéros du tirage du loto et un attribut *complementaire* qui désignera le numéro complémentaire.
2. Définir la méthode ___str___ qui renverra une chaîne de caractères des numéros du loto.
3. Définir la méthode ___contains___ qui vérifiera si le numéro donné en argument est dans les 6 numéros du loto.
4. Définir la méthode *est_gagnant* qui possède deux arguments : une liste d'entiers et un entier. Elle renverra True si le tirage correspond exactement à la proposition.

Exercice 5 : On définit une classe *Fraction* pour représenter un nombre rationnel. Cette classe possède deux attributs *numerateur* et *denominateur*. Le dénominateur doit être strictement positif.

Christophe Viroulaud

1. Écrire le constructeur de cette classe. Il doit lever une ValueError si le dénominateur n'est pas strictement positif.
2. Définir la méthode ___*str*___ qui renvoie une chaîne de caractère de la forme "12 / 7" ou simplement "12" si le dénominateur est égal à 1.
3. Définir les méthodes ___*eq*___ et ___*lt*___ qui reçoivent une deuxième fraction en argument et renvoient True si la première fraction représente respectivement un nombre égal ou strictement inférieur à la deuxième fraction.
4. Définir les méthodes ___*add*___ et ___*mul*___ qui reçoivent une deuxième fraction en argument et renvoient une nouvelle fraction représentant respectivement la somme et le produit des deux fractions.

Exercice 6 : On définit une classe *Date* pour représenter une date avec trois attributs *jour, mois* et *annee*.

1. Écrire son constructeur.
2. Définir la méthode ___*str*___ qui renvoie une chaîne de la forme "8 mai 1945".
3. Définir la méthode ___*lt*___ qui permet de comparer deux dates.
4. Tester ces méthodes.

Christophe Viroulaud