Exercice 1 :

```
class Livre:

    def __init__(self, t: str, a: str, p: int):
        self.titre = t
        self.auteur = a
        self.prix = p

    def get_titre(self) -> str:
        return self.titre

    def set_titre(self, t: str) -> None:
        self.titre = t

    def afficher(self) -> str:
        # return "{} est écrit par {}.".format(self.titre, self.
            auteur)
        return f"{self.titre} est écrit par {self.auteur}."

livre1 = Livre("La guerre des mondes", "Herbert Wells", 7)
print(livre1.afficher())
```

Exercice 2 :

```
class Rectangle:

    def __init__(self, L: float, l: float):
        self.longueur = L
        self.largeur = l

    def get_longueur(self) -> float:
        return self.longueur

    def get_largeur(self) -> float:
        return self.largeur

    def perimetre(self) -> float:
        return (self.longueur + self.largeur)*2

    def aire(self) -> float:
        return round(self.longueur * self.largeur, 2)

    def est_carre(self) -> bool:
        return self.longueur == self.largeur

rect = Rectangle(5.3, 2.8)
print(rect.aire())
print(rect.est_carre())
```

Exercice 3 :

```
class Complexe:
```


Christophe Viroulaud

```
2
3     def __init__(self, re: float, im: float):
4         self.reel = re
5         self.imaginaire = im
6
7     def get_reel(self) -> float:
8         return self.reel
9
10    def get_imaginaire(self) -> float:
11        return self.imaginaire
12
13    def addition(self, c):
14        # nous effectuerons du typing uniquement pour types de
              base
15        return (self.reel + c.reel, self.imaginaire + c.imaginaire
              )
16
17    def afficher(self) -> str:
18        return f"{self.reel} + i*{self.imaginaire}"
19
20 z = Complexe(3, 5)
21 print(z.afficher())
22 print(z.addition(Complexe(8, 2)))
```

Exercice 4 :

```
1  class Loto:
2
3      def __init__(self, num: list, c: int):
4          self.numeros = num
5          self.complementaire = c
6
7      def __contains__(self, n):
8          # implémente l'opérateur in pour cet objet
9          return n in self.numeros
10
11     def __str__(self):
12         #e st appelé quand on veut afficher l'objet
13         return str(self.complementaire)
14
15 tirage = Loto([31,32,53,17,45,36],7)
16 print(tirage) # __str__ est appelée
17 print(5 in tirage) # __contains__ a redéfini in
```

Exercice 5 :

```
1  class Fraction:
2
3      def __init__(self, num: int, den: int):
4          if den < 0:
5              raise ValueError(f"{den} doit être strictement positif
                   .")
```

```
6         self.numerateur = num
7         self.denominateur = den
8
9     def __eq__(self, f):
10        return self.numerateur * f.denominateur == self.
              denominateur * f.numerateur
11
12    def __lt__(self, f):
13        return self.numerateur * f.denominateur < self.
              denominateur * f.numerateur
14
15    def __add__(self, f):
16        return Fraction(self.numerateur * f.denominateur + f.
              numerateur * self.denominateur,
17                        self.denominateur * f.denominateur)
18
19    def __mul__(self, f):
20        return Fraction(self.numerateur * f.numerateur, self.
              denominateur * f.denominateur)
21
22    def __str__(self):
23        if self.denominateur == 1:
24            return str(self.numerateur)
25        else:
26            return f"{self.numerateur}/{self.denominateur}"
27
28 f = Fraction(12,7)
29 print(f)
30 print(f == Fraction(24, 14))
31 print(f + Fraction(2, 3))
```

Exercice 6 :

```
1  class Date:
2
3      # on définit une variable interne àla classe
4      nom_mois = ["janvier", "février", "mars", "avril", "mai", "
          juin",
5                  "juillet", "août", "septembre", "octobre", "
                     novembre", "décembre"]
6
7      def __init__(self, j: int, m: int, a: int):
8          self.jour = j
9          self.mois = m
10         self.annee = a
11
12     def __str__(self):
13         return f"{self.jour} / {Date.nom_mois[self.mois - 1]} / {
              self.annee}"
14
15     def __lt__(self, d):
16         # Le \ permet d'écrire sur plusieurs lignes
```


Christophe Viroulaud

```
17          # and est prioritaire devant or
18          return self.annee < d.annee or \
19              self.annee == d.annee and (self.mois < d.mois or \
20                                         self.mois == d.mois and self.
                                              jour < d.jour)
21
22  d = Date(8, 12, 1977)
23  print(d)
24  print(d < Date(9,12,1977))
```


Christophe Viroulaud