

**Exercice 1**

1) Créer une classe `Eleve` dans laquelle chaque objet possède trois attributs : `nom`, `prenom` et `age`

2) Créer l'objet `bill` correspondant aux attributs suivants :

nom	prenom	age
GATES	Bill	64

3) Créer un objet `moi` correspondant à votre identité.

4) Ajouter une méthode `__str__(self)` qui retourne une chaîne de caractères, de telle sorte que `bill.__str__()` retourne : `'Bill GATES, 64 ans'`

5) Tester cette méthode en exécutant la commande `print(moi)`

**Exercice 2**

1) Créer une classe `Fraction` dans laquelle chaque objet possède deux attributs : `num`, `denom` qui sont des entiers représentant le numérateur et le dénominateur d'une fraction.

2) On exige de plus que le dénominateur soit strictement positif : modifier le constructeur de la classe `Fraction` pour qu'une exception `ValueError` soit levée si cette condition n'est pas vérifiée.

3) Ajouter une méthode `__str__` qui retourne un texte correspondant à la fraction, en gérant le cas particulier où le dénominateur vaut 1.

Ainsi :

- `print(Fraction(35, 7))` devra afficher `35/7`
- `print(Fraction(3, 1))` devra afficher `3`

4) Ajouter une méthode `__eq__(self, v)` telle que retourne `True` si les deux objets correspondent à des fractions égales.

=> Penser à tester cette méthode

5) Ajouter une méthode `multiplie(self, v)` telle que retourne une nouvelle `Fraction` égale au produit des deux `Fractions` données en argument.

=> Penser à tester cette méthode

**5bis)** Renommer cette méthode : `__mul__` En quoi cela facilite-t-il les tests ?

6) Ajouter une méthode `additionne(self, v)` telle que retourne une nouvelle `Fraction` égale à la somme des deux `Fractions` données en argument.

=> Penser à tester cette méthode

**6bis)** on pourra renommer cette méthode : `__add__` . En quoi cela facilite-t-il les tests ?

Questions Bonus : s'assurer que le résultat d'une addition ou multiplication est toujours sous forme réduite.

**Exercice 3**

1) Créer une classe `Date` dans laquelle chaque objet possède trois attributs : `jour`, `mois`, `annee` qui sont des entiers.

2) Ajouter une méthode `__eq__(self, v)` telle que retourne `True` si les deux objets correspondent à des dates identiques.

=> Penser à tester cette méthode

3) Ajouter une méthode `anterieure(self, v)` telle que retourne `True` si la première date est strictement antérieure à la seconde

=> Penser à tester cette méthode

3bis) renommer cette méthode : `__lt__` En quoi cela facilite-t-il les tests.