

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# fichier: entier.py
# version: 0.5.0
# auteur: Pascal CHAUVIN
# date: 2014/10/28
#
# (tous les symboles non internationaux sont volontairement omis)
#
```

```
class entier(object):
    """ nombre entier relatif """

    def __init__(self, valeur =0, valide =True):
        """ constructeur """
        if not valide:
            valeur = 0

        self.__valeur = valeur
        self.__valide = valide

    def __repr__(self):
        """ _ """
        return "[entier:\n__valeur={0},\n__valeur={1}\n]\n"\
            .format(self.__valeur, self.__valide)

    def __str__(self):
        """ _ """
        return str(self.__valeur)

    def joli(self):
        """ _ """
        return self.__str__()

    def est_valide(self):
        """ accesseur """
        return self.__valide

    def lire_valeur(self):
        """ accesseur """
        return self.__valeur

    def __add__(self, autre):
        """ addition """
        if isinstance(autre, entier):
            if (self.__valide) and (autre.__valide):
                a = self.__valeur
                b = autre.__valeur
                return entier(a + b)

        return entier(0, False)

    def __neg__(self):
```

```
    """ oppose """
    if self.__valide:
        return entier(-self.__valeur)

    return entier(0, False)

def oppose(self):
    """ oppose """
    return self.__neg__()

def __sub__(self, autre):
    """ soustraction """
    return (self + (-autre))

def __mul__(self, autre):
    """ multiplication """
    if isinstance(autre, entier):
        if (self.__valide) and (autre.__valide):
            a = self.__valeur
            b = autre.__valeur
            return entier(a * b)

    return entier(0, False)

def __pow__(self, autre):
    """ exponentiation (exposant entier naturel) """
    if isinstance(autre, entier):
        if (self.__valide) and (autre.__valide):
            a = self.__valeur
            n = autre.__valeur

            if n == 0 and a == 0:
                return entier(0, False)

            if n < 0 and abs(a) != 1:
                return entier(0, False)

            return entier(a ** n)

    return entier(0, False)

def __truediv__(self, autre):
    """ division (si quotient entier exact) """
    if isinstance(autre, entier):
        if (self.__valide) and (autre.__valide):
            a = self.__valeur
            b = autre.__valeur

            if b == 0:
                return entier(0, False)

            if a % b == 0:
                return entier(a // b)

    return entier(0, False)
```

```
def est_zero(self):
    """ _ """
    if self.__valide:
        return (self.__valeur == 0)

    return False

def est_un(self):
    """ _ """
    if self.__valide:
        return (self.__valeur == 1)

    return False

def pgcd_naturels(a, b):
    """ plus grand commun diviseur de deux entiers naturels """
    while b != 0:
        a, b = b, a % b
    return a

def pgcd_entiers(a, b):
    """ plus grand commun diviseur de deux entiers """
    return pgcd_naturels(abs(a), abs(b))

def pgcd_liste(l):
    """ pgcd d'une liste de nombres entiers """
    if len(l) == 0:
        return 1

    if len(l) == 1:
        return abs(l[0])

    g = pgcd_entiers(l[0], l[1])
    if len(l) == 2:
        return g

    for n in l[1:]:
        g = pgcd_entiers(g, n)

    return g

def ppcm_naturels(a, b):
    """ plus petit multiple commun de deux entiers naturels """
    if a == 0 or b == 0:
        return 0
    else:
        return ((a*b) // pgcd_naturels(a, b))

def ppcm_entiers(a, b):
    """ plus grand commun diviseur de deux entiers """
    return ppcm_naturels(abs(a), abs(b))

if __name__ == "__main__":
    pass
```

