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

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

import sys
sys.path.append('../entier_mod')

import entier

class rationnel(object):
  """ nombre entier rationnel """

  def __init__(self, num =0, denom =1, valide =True):
    """ constructeur """
    if (not valide) or (denom == 0):
      num, denom = 0, 1
      valide = False

    num_ok = False

    if isinstance(num, int):
      self.__num = entier.entier(num)
      num_ok = True

    if isinstance(num, entier.entier):
      self.__num = num
      num_ok = num.est_valide()

    denom_ok = False

    if isinstance(denom, int):
      self.__denom = entier.entier(denom)
      denom_ok = True

    if isinstance(denom, entier.entier):
      self.__denom = denom
      denom_ok = denom.est_valide()

    self.__valide = valide and \
      (num_ok and denom_ok) and \
      (not self.__denom.est_zero())

    if self.__valide:
      p = self.__num.lire_valeur()
      q = self.__denom.lire_valeur()

      if q < 0:
        p, q = -p, -q

      n = entier.pgcd_entiers(p, q)

      self.__num = entier.entier(p // n)
      self.__denom = entier.entier(q // n)
    else:
      self.__num = entier.entier(0, False)
      self.__denom = entier.entier(1)



  def __repr__(self):
```

```
    """ - """
    return "[rationnel:\n__num={0},\n__denom={1},\n__valide={2}\n]\n".\
      format(self.__num, self.__denom, self.__valide)



  def __str__(self):
    """ - """
    if self.__denom.lire_valeur() == 1:
      return str(self.__num)
    else:
      return str(self.__num) + "/" + str(self.__denom)



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



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



  def lire_num(self):
    """ accesseur """
    return self.__num



  def lire_denom(self):
    """ accesseur """
    return self.__denom



  def __eq__(self, autre):
    """ addition """
    if isinstance(autre, rationnel):
      if (self.__valide) and (autre.__valide):
        a = self.__num.lire_valeur()
        b = self.__denom.lire_valeur()

        p = autre.__num.lire_valeur()
        q = autre.__denom.lire_valeur()

        return (a*q == b*p)

    return rationnel(0, 1, False)



  def __add__(self, autre):
    """ addition """
    if isinstance(autre, rationnel):
      if (self.__valide) and (autre.__valide):
        a = self.__num.lire_valeur()
        b = self.__denom.lire_valeur()

        p = autre.__num.lire_valeur()
        q = autre.__denom.lire_valeur()

        return rationnel(a*q + b*p, b*q)

    return rationnel(0, 1, False)
```

```
  def __neg__(self):
    """ oppose """
    return rationnel(-self.__num.lire_valeur(), self.__denom.lire_valeur())


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


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

    return rationnel(0, 1, False)


  def __mul__(self, autre):
    """ multiplication """
    if isinstance(autre, rationnel):
      if (self.__valide) and (autre.__valide):
        a = self.__num.lire_valeur()
        b = self.__denom.lire_valeur()

        p = autre.__num.lire_valeur()
        q = autre.__denom.lire_valeur()

        return rationnel(a*p, b*q)

    return rationnel(0, 1, False)


  def __pow__(self, autre):
    """ exponentiation (par un entier relatif, si definie) """
    if isinstance(autre, rationnel):
      if (self.__valide) and (autre.__valide):
        a = self.__num.lire_valeur()
        b = self.__denom.lire_valeur()

        p = autre.__num.lire_valeur()
        q = autre.__denom.lire_valeur()

        if p < 0:
          a, b = b, a
          p = -p

        if q != 1:
          return rationnel()

        if p == 0 and a == 0:
          return rationnel()

        return rationnel(a**p, b**p)

    return rationnel(0, 1, False)


  def __truediv__(self, autre):
    """ division """
```

```
    if isinstance(autre, rationnel):
      if (self.__valide) and (autre.__valide):
        a = self.__num.lire_valeur()
        b = self.__denom.lire_valeur()

        p = autre.__num.lire_valeur()
        q = autre.__denom.lire_valeur()

        if p == 0:
          return rationnel(0, 1, False)

        return rationnel(a*q, b*p)

    return rationnel(0, 1, False)



  def est_zero(self):
    """ - """
    return self.__num.est_zero()



  def est_un(self):
    """ - """
#   return (self.__num.est_un()) and (self.__denom.est_un())
    return (self.__num.lire_valeur() == self.__denom.lire_valeur())



  def est_entier(self):
    """ - """
    return (self.__denom.est_un())



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