```
#!/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
```

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

#
# fichier: rationnel_tests.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 rationnel

import entier

def test_unitaire_0(visible =False):
  print("*** rationnel: test_unitaire_0 ***")

  a = rationnel.rationnel(-400, -50)
  if visible:
    print(a)
    print(repr(a))

  ok = (a.est_valide() and \
    a.lire_num().lire_valeur() == 8 and a.lire_denom().lire_valeur() == 1)

  return ok



def test_unitaire_1(visible =False):
  print("*** rationnel: test_unitaire_1 ***")

  a = rationnel.rationnel(2, 0)

  ok = (not a.est_valide())

  return ok



def test_unitaire_2(visible =False):
  print("*** rationnel: test_unitaire_2 ***")

  a = rationnel.rationnel(-400, 23, False)

  ok = (not a.est_valide())

  return ok



def test_unitaire_3(visible =False):
  print("*** rationnel: test_unitaire_3 ***")

  a = rationnel.rationnel(-40, -200)
  if visible:
    print(a)
    print(repr(a))

  ok = (a.est_valide() and \
    (a.lire_num().lire_valeur() == 1) and (a.lire_denom().lire_valeur() == 5))

  return ok
```

```
def test_unitaire_4(visible =False):
  print("*** rationnel: test_unitaire_4 ***")

  a = rationnel.rationnel(-2)

  ok = (a.est_valide() and \
    (a.lire_num().lire_valeur() == -2) and (a.lire_denom().lire_valeur() == 1))

  return ok



def test_unitaire_5(visible =False):
  print("*** rationnel: test_unitaire_5 ***")

  a = rationnel.rationnel("12", 5)

  ok = (not a.est_valide())

  return ok



def test_unitaire_6(visible =False):
  print("*** rationnel: test_unitaire_6 ***")

  a = rationnel.rationnel(entier.entier(-15), entier.entier(-40))

  ok = (a.est_valide() and \
    (a.lire_num().lire_valeur() == 3) and (a.lire_denom().lire_valeur() == 8))

  return ok



def test_unitaire_7(visible =False):
  print("*** rationnel: test_unitaire_7 ***")

  a = rationnel.rationnel(entier.entier(5), -4)
  b = rationnel.rationnel(2, 3)
  x = a + b

  ok = (x.est_valide() and \
    x.lire_num().lire_valeur() == -7 and x.lire_denom().lire_valeur() == 12)

  return ok



def test_unitaire_8(visible =False):
  print("*** rationnel: test_unitaire_8 ***")

  a = rationnel.rationnel(entier.entier(5), -4)
  b = rationnel.rationnel(2, 3)
  x = a * b

  ok = (x.est_valide() and \
    x.lire_num().lire_valeur() == -5 and x.lire_denom().lire_valeur() == 6)

  return ok



def test_unitaire_9(visible =False):
```

```
  print("*** rationnel: test_unitaire_9 ***")

  a = rationnel.rationnel(entier.entier(5), -4)
  b = rationnel.rationnel(2, 3)
  x = a / b

  ok = (x.est_valide() and \
    x.lire_num().lire_valeur() == -15 and x.lire_denom().lire_valeur() == 8)

  return ok



def test_unitaire_10(visible =False):
  print("*** rationnel: test_unitaire_10 ***")

  a = rationnel.rationnel(20, -14)
  n = rationnel.rationnel(24, 6)
  x = a ** n

  ok = (x.est_valide() and \
    x.lire_num().lire_valeur() == 10000 and x.lire_denom().lire_valeur() == 2401)

  return ok



def test_unitaire_11(visible =False):
  print("*** rationnel: test_unitaire_11 ***")

  a = rationnel.rationnel(20, -14)
  n = rationnel.rationnel(6, -2)
  x = a ** n

  ok = (x.est_valide() and \
    x.lire_num().lire_valeur() == -343 and x.lire_denom().lire_valeur() == 1000)

  return ok



def test_unitaire_12(visible =False):
  print("*** rationnel: test_unitaire_12 ***")

  a = rationnel.rationnel(5, -4)
  b = rationnel.rationnel()
  x = a / b

  ok = (not x.est_valide())

  return ok



def test_unitaire_13(visible =False):
  print("*** rationnel: test_unitaire_13 ***")

  a = rationnel.rationnel(5, -4)
  b = rationnel.rationnel(1, 2)
  c = rationnel.rationnel(3)
  x = (a + b) ** c
  if visible: print(x)

  ok = x.est_valide()

  return ok
```

```
def test_unitaire_14(visible =False):
  print("*** rationnel: test_unitaire_14 ***")

  a = rationnel.rationnel(5, -4)
  b = rationnel.rationnel(1, 2, False)
  c = rationnel.rationnel(3)
  x = (a + b) ** c

  ok = (not x.est_valide())

  return ok



def test_unitaire_15(visible =False):
  print("*** rationnel: test_unitaire_15 ***")

  a = rationnel.rationnel(0, False)
  b = rationnel.rationnel(0, 1, False)

  ok = (a.lire_num().lire_valeur() == b.lire_num().lire_valeur()) and \
    (a.lire_denom().lire_valeur() == b.lire_denom().lire_valeur())

  return ok



def test_unitaire_16(visible =False):
  print("*** rationnel: test_unitaire_16 ***")

  a = rationnel.rationnel(2, -4)
  b = rationnel.rationnel(1, 2)
  c = rationnel.rationnel(-3)
  x = (a + b) ** c

  ok = (not x.est_valide())

  return ok



def test_unitaire_17(visible =False):
  print("*** rationnel: test_unitaire_17 ***")

  a = rationnel.rationnel(2, 4)
  b = rationnel.rationnel(1, 2)

  ok = (a == b)

  return ok



def test_unitaire_18(visible =False):
  print("*** rationnel: test_unitaire_18 ***")

  a = rationnel.rationnel(entier.entier(5), -4)
  b = rationnel.rationnel(0, 3)
  x = a / b

  ok = (not x.est_valide())

  return ok



def test_unitaire_19(visible =False):
```

```
  print("*** rationnel: test_unitaire_19 ***")

  a = rationnel.rationnel(entier.entier(5), -4)
  b = rationnel.rationnel(2, 3, False)
  x = a + b

  ok = (not x.est_valide())

  return ok



def test_unitaire_(visible =False):
  print("*** fraction: test_unitaire_ ***")

  ok = True
  return ok



def tests_unitaires():
  return (
    test_unitaire_0() and \
    test_unitaire_1() and \
    test_unitaire_2() and \
    test_unitaire_3() and \
    test_unitaire_4() and \
    test_unitaire_5() and \
    test_unitaire_6() and \
    test_unitaire_7() and \
    test_unitaire_8() and \
    test_unitaire_9() and \
    test_unitaire_10() and \
    test_unitaire_11() and \
    test_unitaire_12() and \
    test_unitaire_13(True) and \
    test_unitaire_14() and \
    test_unitaire_15() and \
    test_unitaire_16() and \
    test_unitaire_17() and \
    test_unitaire_18() and \
    test_unitaire_19()
  )



if __name__ == "__main__":
  ok = tests_unitaires()
  if ok:
    print("*** rationnel: tests unitaires OK ***")
```