```python
#!/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
```

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

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

import entier

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

  a = entier.entier(5)
  if visible:
    print(a)
    print(repr(a))

  ok = True

  return ok



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

  a = entier.entier(5, 0 == 1)

  ok = (not a.est_valide())

  return ok



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

  a = entier.entier(5, 0 == 1)

  ok = (not a.est_valide())

  return ok



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

  a = entier.entier(5)
  b = entier.entier(6)
  x = a + b

  ok = (x.est_valide() and x.lire_valeur() == 11)

  return ok



def test_unitaire_4(visible =False):
  print("*** entier: test_unitaire_4 ***")

  a = entier.entier(5)
  b = entier.entier(6, False)
```

```
  x = a + b

  ok = (x.est_valide() == False and x.lire_valeur() == 0)

  return ok



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

  a = entier.entier(5)
  b = entier.entier(6)
  x = a - b

  ok = (x.est_valide() and x.lire_valeur() == -1)

  return ok



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

  a = entier.entier(5)
  x = -a
  y = a.oppose()

  ok = ((x.est_valide() and x.lire_valeur() == -5) and \
    (y.est_valide() and y.lire_valeur() == -5))

  return ok



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

  a = entier.entier(5, False)
  x = -a

  ok = (x.est_valide() == False and x.lire_valeur() == 0)

  return ok



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

  a = entier.entier(5)
  n = entier.entier(6)
  x = a ** n

  ok = (x.est_valide() and x.lire_valeur() == 15625)

  return ok



def test_unitaire_9(visible =False):
  print("*** entier: test_unitaire_9 ***")

  a = entier.entier(5)
  n = entier.entier(-6)
  x = a ** n
```

```
  ok = (not x.est_valide())

  return ok



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

  a = entier.entier(0)
  n = entier.entier(6)
  x = a ** n

  ok = (x.est_valide() and x.lire_valeur() == 0)

  return ok



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

  a = entier.entier(0)
  n = entier.entier(0)
  x = a ** n

  ok = (not x.est_valide())

  return ok



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

  a = entier.entier(-1)
  n = entier.entier(-10)
  x = a ** n

  ok = (x.est_valide() and x.lire_valeur() == 1)

  return ok



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

  a = entier.entier(-54)
  b = entier.entier(6)
  x = a / b

  ok = (x.est_valide() and x.lire_valeur() == -9)

  return ok



def test_unitaire_14(visible =False):
  print("*** entier: test_unitaire_14 ***")

  a = entier.entier(-55)
  b = entier.entier(6)
  x = a / b

  ok = (x.est_valide() == False)

  return ok
```

```python
def test_unitaire_15(visible =False):
  print("*** entier: test_unitaire_15 ***")

  a = entier.entier(-55)
  b = entier.entier()
  x = a / b

  ok = (x.est_valide() == False)

  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() and \
    test_unitaire_14() and \
    test_unitaire_15()
  )



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