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

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

import fraction as frac

import monome as mo
import polynome as po
import rationnel as ra

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

  f = frac.fraction()
  if visible: print(f)

  ok = f.est_valide()
  return ok



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

  f = frac.fraction(po.polynome_nul(), po.polynome_un(), False)
  if visible: print(f)

  ok = not f.est_valide()
  return ok



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

  a = po.polynome(mo.monome(ra.rationnel(1), "a"))
  b = po.polynome(mo.monome(ra.rationnel(1), "b"))

  c = po.polynome(mo.monome(ra.rationnel(1), "c"))
  d = po.polynome(mo.monome(ra.rationnel(1), "d"))

  f = frac.fraction(a, b)
  g = frac.fraction(c, d)

  r = f + g
  if visible:
    print(r)
    print(r.joli())

  ok = r.est_valide()
  return ok



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

  a = po.polynome(mo.monome(ra.rationnel(1), "a"))

  #
  # on provoque volontairement une erreur
  #
  b = po.polynome(mo.monome(ra.rationnel(1), "b", False))

  c = po.polynome(mo.monome(ra.rationnel(1), "c"))
  d = po.polynome(mo.monome(ra.rationnel(1), "d"))

  f = frac.fraction(a, b)
  g = frac.fraction(c, d)

  r = f + g
  if visible: print(r)

  ok = not r.est_valide()
  return ok



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

  a = po.polynome(mo.monome(ra.rationnel(1), "x"))
  a = a.joindre(mo.monome(ra.rationnel(1)))
# if visible: print(a)

  b = po.polynome(mo.monome(ra.rationnel(1), "x"))
  b = b.joindre(mo.monome(ra.rationnel(3)))
# if visible: print(b)

  c = po.polynome(mo.monome(ra.rationnel(-1), "xx"))
# if visible: print(c)

  d = po.polynome(mo.monome(ra.rationnel(1)))
# if visible: print(d)

  f = frac.fraction(a, b)
  if visible: print(f)

  g = frac.fraction(c, d)
  if visible: print(g)

  r = f + g
  if visible: print(r)

  ok = r.est_valide()
  return ok



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

  a = po.polynome(mo.monome(ra.rationnel(1), "x"))
  a = a.joindre(mo.monome(ra.rationnel(1)))
# if visible: print(a)

  b = po.polynome(mo.monome(ra.rationnel(1), "x"))
  b = b.joindre(mo.monome(ra.rationnel(3)))
# if visible: print(b)

  c = po.polynome(mo.monome(ra.rationnel(-1), "xx"))
# if visible: print(c)

  d = po.polynome(mo.monome(ra.rationnel(1)))
# if visible: print(d)

  f = frac.fraction(a, b)
# if visible: print(f)
```

```
  g = frac.fraction(c, d)
# if visible: print(g)

  r = f - g
  if visible: print(r)

  ok = r.est_valide()
  return ok



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

  a = po.polynome(mo.monome(ra.rationnel(1), "x"))
  a = a.joindre(mo.monome(ra.rationnel(1)))
# if visible: print(a)

  b = po.polynome(mo.monome(ra.rationnel(1), "x"))
  b = b.joindre(mo.monome(ra.rationnel(3)))
# if visible: print(b)

  c = po.polynome(mo.monome(ra.rationnel(-1), "xx"))
# if visible: print(c)

  d = po.polynome(mo.monome(ra.rationnel(1)))
# if visible: print(d)

  f = frac.fraction(a, b)
# if visible: print(f)

  g = frac.fraction(c, d)
# if visible: print(g)

  r = f * g
  if visible: print(r)

  ok = r.est_valide()
  return ok



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

  a = po.polynome(mo.monome(ra.rationnel(1), "x"))
  a = a.joindre(mo.monome(ra.rationnel(1)))
# if visible: print(a)

  b = po.polynome(mo.monome(ra.rationnel(1), "x"))
  b = b.joindre(mo.monome(ra.rationnel(3)))
# if visible: print(b)

  c = po.polynome(mo.monome(ra.rationnel(-1), "xx"))
# if visible: print(c)

  d = po.polynome(mo.monome(ra.rationnel(1)))
# if visible: print(d)

  f = frac.fraction(a, b)
# if visible: print(f)

  g = frac.fraction(c, d)
# if visible: print(g)

  r = f / g
  if visible: print(r)
  ok = r.est_valide()
  return ok



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

  a = po.polynome(mo.monome(ra.rationnel(1), "a"))
  a = a.joindre(mo.monome(ra.rationnel(1), "b"))
# if visible: print(a)

  b = po.polynome(mo.monome(ra.rationnel(3)))
# if visible: print(b)

  c = po.polynome(mo.monome(ra.rationnel(5)))
# if visible: print(c)

  f = frac.fraction(a, b)
  if visible: print(f)

  g = frac.fraction(c, po.polynome_un())
  if visible: print(g)

  r = f ** g
  if visible:
    print(r)
    print(r.joli())

  ok = (r.lire_num().valuation() == ra.rationnel(1, 243))
  return ok



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

  a = po.polynome(mo.monome(ra.rationnel(2)))
# if visible: print(a)

  b = po.polynome(mo.monome(ra.rationnel(1)))
# if visible: print(b)

  c = po.polynome(mo.monome(ra.rationnel(3)))
# if visible: print(c)

  d = po.polynome(mo.monome(ra.rationnel(4)))
# if visible: print(d)

  f = frac.fraction(a, b)
# if visible: print(f)

  g = frac.fraction(c, d)
# if visible: print(g)

  r = (f + g) + (f - g) + (f * g) + (f / g)
  if visible: print(r)

  ok = (r == ra.rationnel(49, 6))
  return ok



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

  a = po.polynome(mo.monome(ra.rationnel(1), "a"))
  a = a.joindre(mo.monome(ra.rationnel(1), "b"))
# if visible: print(a)

  b = po.polynome(mo.monome(ra.rationnel(3)))
# if visible: print(b)

  f = frac.fraction(a, b)
  if visible: print(f)

  r = f ** 5
  if visible: print(r)

  ok = (r.lire_denom().valuation().lire_num().lire_valeur() == 243)
  return ok



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

  f = frac.fraction_nulle()
  if visible: print("fraction nulle:", f)

  ok1 = f.lire_num().valuation().est_zero()

  f = frac.fraction_un()
  if visible: print("fraction unite:", f)
  ok2 = f.lire_num().valuation().est_un()

  f = frac.fraction_err()
  if visible: print("fraction erreur:", f)
  ok3 = not f.est_valide()

  ok = ok1 and ok2 and ok3
  return ok



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

  a = po.polynome(mo.monome(ra.rationnel(3)))
  if visible: print(a)

  b = po.polynome(mo.monome(ra.rationnel(19)))
  if visible: print(b)

  b = b.joindre(mo.monome(ra.rationnel(1)))
  if visible: print(b)

  b = b.joindre(mo.monome(ra.rationnel(-20)))
  if visible: print(b)

  f = frac.fraction(a, b)
  if visible: print(f)

  ok = (not f.est_valide())
  return ok



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

  ok = True
  return ok
```

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

  ok = True
  return ok



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

  ok = True
  return ok



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

  ok = True
  return ok



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

  ok = True
  return ok



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

  ok = True
  return ok



def test_unitaire_19(visible =False):
  print("*** fraction: test_unitaire_19 ***")

  ok = True
  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(True) and \
    test_unitaire_3() and \
    test_unitaire_4(True) and \
    test_unitaire_5(True) and \
    test_unitaire_6(True) and \
    test_unitaire_7(True) and \
    test_unitaire_8(True) and \
    test_unitaire_9() and \
    test_unitaire_10() and \
    test_unitaire_11(True) and \
    test_unitaire_12(True) and \
    test_unitaire_13() 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("*** fraction: tests unitaires OK ***")
```