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

#
# fichier: monome.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('./joli_mod')
sys.path.append('../utile_mod')

import joli
import rationnel

import utile

INDET_DEG_0 = "?"

class monome(object):

  def __init__(self, coeff =rationnel.rationnel(), indet =INDET_DEG_0, valide =True):
    """ - """
    if not valide:
      coeff = rationnel.rationnel(0, 1, False)
      indet = INDET_DEG_0
    else:
      if len(indet) == 0:
        indet =INDET_DEG_0
        valide = True
      else:
        t = utile.en_ordre_alphabetique(indet)
        valide = not utile.contient_erreur(t)

        if valide:
          indet = t
        else:
          indet = INDET_DEG_0

    if coeff.est_zero():
      indet = INDET_DEG_0

    self.__valide = valide
    self.__coeff = coeff
    self.__indet = indet



  def __repr__(self):
    """ - """
    return "[monome:\n__coeff={0},\n__indet={1},\n__valide={2}\n]\n".\
      format(self.__coeff, self.__indet, self.__valide)



  def __str__(self):
    """ - """
    t = ""
    if self.__coeff.lire_num().lire_valeur() >= 0:
      t = str(self.__coeff)
    else:
      t = "(" + str(self.__coeff) + ")"

    if self.__indet.endswith(INDET_DEG_0):
      return t

    if self.__coeff == rationnel.rationnel(1):
      return str(self.__indet)
    else:
      return t + " * " + str(self.__indet)



  def joli(self):
    """ - """
    t = ""
    if self.__coeff.lire_num().lire_valeur() >= 0:
      t = str(self.__coeff)
    else:
      t = "(" + str(self.__coeff) + ")"

    if self.__indet.endswith(INDET_DEG_0):
      return t

    indet = joli.format_indet(self.__indet)

    if self.__coeff == rationnel.rationnel(1):
      return str(indet)
    else:
      return t + " * " + str(indet)



  def __eq__(self, autre):
    """ - """
    return (self.__indet is autre.__indet)



  def __lt__(self, autre):
    """ ordre sur les indeterminees (polynome multivarie) """
#   return (self.__indet < autre.__indet)
    a = self.__indet
    if a.startswith(INDET_DEG_0):
      a = ""

    b = autre.__indet
    if b.startswith(INDET_DEG_0):
      b = ""

    if len(a) != len(b):
      return (a < b)
    else:
      return (a > b)



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



  def lire_coeff(self):
    """ - """
    return self.__coeff



  def fixer_coeff(self, c):
    """ - """
    self.__coeff = c
  def lire_indet(self):
    """ - """
    return self.__indet


  def fixer_indet(self, i):
    """ - """
    self.__indet = i


  def est_degre_nul(self):
    """ - """
    return self.__indet.startswith(INDET_DEG_0)


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