```
'''
IPEIN
CHP 2 :  POO
Exercice : 8
'''

from math import sqrt
#Q1
class vect2d:
   def __init__(self, x=0, y=0):
      self.x = x
      self.y = y
   # Q2
   def add(self,other):
       assert type(other)== vect2d
       x = self.x + other.x
       y = self.y + other.y
       v = vect2d(x,y)
       return v

   #Q3
   def mul_ext(self,k):
       return vect2d(self.x * k , self.y *k)

   # Q4
   def zoom(self,k):
       self.x = self.x * k
       self.y = self.y * k

   # Q5
   def prodscal(self, other):
       if type(other) != vect2d:
           raise TypeError("Messae d'erreur")
       else:
           # norme(v1)  *norme(v2)  * cos(
           return self.x * other.x + self.y * other.y

   # Q6
   def norme(self):
       #sqrt(x**2+y**2)
       return sqrt(self.x **2 + self.y **2)


   #Q7
   def __add__(self, other):
       return self.add(other)

   def __str__(self):
       #role:  convertir un objet de type vect2d
       #       en objet de type str
       # doit retourner une seule chaine
       # print("appel à vectd_str__")
       return "vect2d(x={},y={})".format(self.x,self.y)

   def __repr__(self):
       print("appel à vect2d.__repr__")
       return str(self)
u = vect2d(3,-2)
v = vect2d(4,1)
v3 = v1.add(v2)
w = v1 + v2
assert (u + w).prodscal(v) == u.prodscal(v) + w.prodscal(v)
assert u.mul_ext(5).prodscal(v) == 5 * u.prodscal(v)
# 5 u => doit appler u.mul_ext(5)
# 5 * u ==> int.__mul__(vect2d)
# u * 5 ==> vect2d.__mul__(nombre)
print(v3) # il faut définir la méthode spéciale __str__ ou __repr__
```