DS 02 (suite et fin) corrigé

Exercice 1 - Analyse de code

Voici ce qui se passe lors de l'utilisation du script :

Code Python

```
from Alarm import Alarme

alarme1 = Alarme("Loritz", "971971971", False) # aucun événement
alarme2 = Alarme("Poincaré", "971971971", True) # idem
alarme1.intrusion() # date est interrogé et renvoie 0000 mais rien n'est consigné
alarme1.activer() # '0001 Activation ' est enregistré dans le journal de alarme1
alarme1.intrusion() # '0002 Intrusion envoi sms au 971971971' aussi
# '971971971 Loritz : 0002 Intrusion' est envoyé par SMS par alarme1
alarme1.desactiver() # '0003 Désactivation ' est enregistré dans le journal de alarme1
alarme2.intrusion() # '0004 Intrusion envoi sms au 971971971' est enregistré dans le
    journal de alarme2
# '971971971 Poincaré : 0004 Intrusion'  est envoyé par SMS par alarme1
```

Donc les SMS envoyés sont

- `'971971971 Loritz : 0002 Intrusion'`
- `'971971971 Poincaré : 0004 Intrusion'`

Et en définitive, `alarme1.journal` vaut :

`['0001 Activation ', '0002 Intrusion envoi sms au 971971971', '0003 Désactivation ']`

Dans le script donné en énoncé, on constate que la méthode `intrusion` est mal codée : on n'enregistre les évènements que lorsque l'alarme est active. Pour remédier à cela, il suffit de désindenter la ligne 20 : ainsi on consigne l'événement dans le journal quoi qu'il advienne.

Pour en plus gérer les envois de SMS échoués on aboutit au code suivant pour la méthode `intrusion` :

Code Python

```
def intrusion(self):
    evenement = date() + " Intrusion"
    if self.active:
        sms = self.lieu + ' : ' + evenement
        if envoie_sms(self.telephone, sms):
            evenement = evenement + " envoi sms au " + self.telephone
        else:
            evenement += " envoi au " + self.telephone + "échoué"
    self.journal.append(evenement)
```

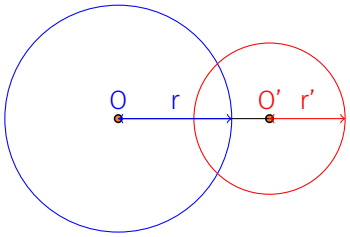
Et pour vider le journal voici la méthode la plus simple :

Code Python

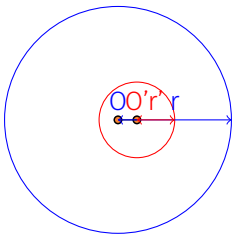
```
def efface_journal(self):
    self.journal = []
```

Exercice 2 - Cercles

La principale difficulté est se déterminer à quelle condition deux cercles se chevauchent (c'est-à-dire ont un point d'intersection) d'une part, et d'autre part à quelle condition un cercle est contenu dans l'autre.



Quand les cercles se chevauchent on a $OO' < r + r'$ mais il ne faut pas non plus que OO' soit trop petit :



Lorsque $OO' < r - r'$ les cercles ne se chevauchent plus mais le grand contient le petit
On construit donc notre classe en faisant bien attention à ce que ces deux cas s'excluent :

Code Python

```
from math import sqrt


def distance(xa, ya, xb, yb):  # La distance usuelle (on ne demande pas de la coder)
    return sqrt((xb - xa) ** 2 + (yb - ya) ** 2)


class Cercle:

    def __init__(self, x: float, y: float, rayon: float):
        self.x = x
        self.y = y
        self.rayon = rayon

    def dilate(self, facteur):  # on multiplie le rayon par un facteur >0
        self.rayon *= facteur

    def decale(self, dx: float, dy: float):  # on translate
        self.x += dx
        self.y += dy
```

```
    def contient(self, other):
        return distance(self.x, self.y, other.x, other.y) <= self.rayon - other.rayon

    def chevauche(self, other):
        return not self.contient(other) and distance(self.x, self.y, other.x, other.y)
            <= self.rayon + other.rayon

    def __str__(self):
        return f"Cercle : centre({self.x}, {self.y}) rayon {self.rayon}."
```