Proposition de correction

Exercice 1

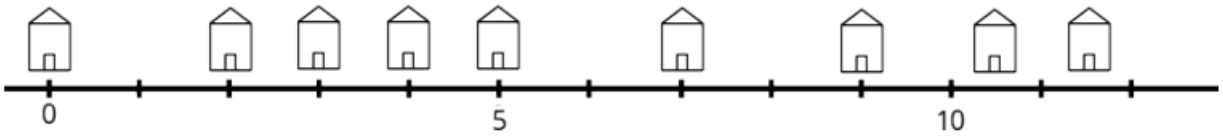
Q1

```
m1 = Maison(1)
m2 = Maison(3.5)
```

Q2

```
a = Antenne(2.5, 1)
```

Q3



Q4

```
def creation_rue(pos : list) -> list:
    """
    @param pos -- liste contenant des nombres correspondant aux positions des maisons.
    @return une liste d'objets de type Maison.
    """
    pos.sort()
    maisons = []
    for p in pos:
        m = Maison(p)
        maisons.append(m)
    return maisons
```

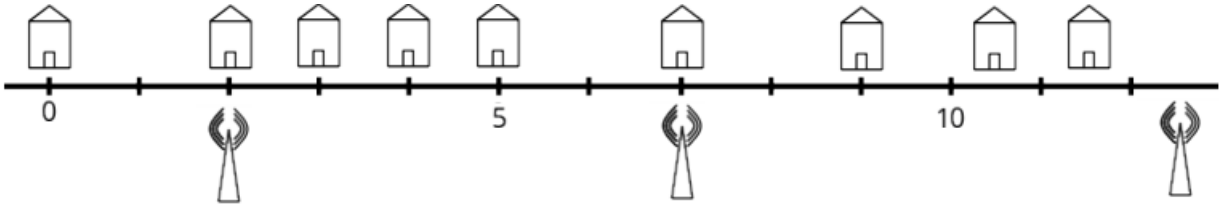
Q5

```
def couvre(self, maison : Maison) -> bool:
    return abs(self.get_pos_antenne() - maison.get_pos_maison()) <= self.get_rayon()
```

Q6

```
>>> maisons = creation_rue([0, 2, 3, 4, 5, 7, 9, 10.5, 11.5])
liste des maisons positionnées comme indiqué en Q3
>>> antennes = strategie_1(maisons, 2)
liste des antennes qui couvrent les maisons
>>> print([a.get_pos_antenne() for a in antennes])
[0, 3, 7, 10.5]  # liste des positions des antennes qui couvrent les maisons
```

Q7



Q8

```
def strategie_2(maisons : list, rayon : float) -> list:
    """ place les antennes de façon optimum
    @param maisons -- une liste de maisons
    @param le rayon d'action des antennes
    @return une liste d'antennes
    @remark          la liste des maisons doit être triée par positions croissantes
    """
    antennes = [Antenne(maisons[0].get_pos_maison()+rayon, rayon)]
    for m in maisons[1:]:
        if not antennes[-1].couvre(m):
            antennes.append(Antenne(m.get_pos_maison()+rayon, rayon))
    return antennes
```

Q9

- strategie_1 : O(n), coût linéaire
- strategie_2 : O(n), coût linéaire