Simulation de coccinelles et de pucerons

Sujet n°7 — Corrigé

Question 1 *Population initiale et simulation sur 5 jours*

```
from coccinelles import Coccinelle, evolution

population = [
    Coccinelle("femelle", 10, 2),
    Coccinelle("femelle", 10, 2),
    Coccinelle("male", 10, 2),
]
nb_proies = 200

for jour in range(1, 6):
    population, nb_proies = evolution(population, nb_proies)
    print(f"Jour {jour} : {len(population)} coccinelle(s), {nb_proies} puceron(s)")
```

Résultats indicatifs (graine fixée à 42) :

```
Jour 1 : 7 coccinelle(s), 184 puceron(s)
Jour 2 : 7 coccinelle(s), 122 puceron(s)
Jour 3 : 11 coccinelle(s), 67 puceron(s)
Jour 4 : 11 coccinelle(s), 16 puceron(s)
Jour 5 : 11 coccinelle(s), 0 puceron(s)
```

Question 2 *Simulation sur 30 jours au plus*

```
def simulation_simple(population, nb_proies):
    for jour in range(1, 31):
        population, nb_proies = evolution(population, nb_proies)
        if len(population) == 0 or nb_proies == 0:
            return (len(population), nb_proies, jour)
    return (len(population), nb_proies, 30)
```

Test (résultat indicatif) :

```
>>> population = [Coccinelle("femelle", 10, 2),
...               Coccinelle("femelle", 10, 2),
...               Coccinelle("male", 10, 2)]
>>> simulation_simple(population, 1000)
(63, 0, 10)
```

Question 3 *Documentation de la méthode* `chasser`

```
def chasser(self, nb_proies, nb_coccinelles):
    """
    Simule la chasse d'une coccinelle sur une journée.

    Paramètres :
        nb_proies      -- nombre total de pucerons disponibles (int)
        nb_coccinelles -- nombre total de coccinelles en vie (int)
```

```
    La quantité consommée dépend du ratio proies/coccinelle :
        - ratio > 20  : entre 12 et 20 pucerons
        - ratio > 10  : entre  8 et 15 pucerons
        - sinon       : entre  3 et  8 pucerons
    La consommation est plafonnée au nombre de pucerons restants.

    Si la coccinelle consomme au moins 10 pucerons, son niveau de
    nutrition augmente de 1 ; sinon il diminue de 1 (minimum 0).

    Renvoie le nombre de pucerons restants après la chasse (int).
    """
    if nb_coccinelles == 0:
        # Aucune coccinelle ne chasse : les proies sont inchangées
        return nb_proies

    # Ratio de proies disponibles par coccinelle
    proies_par_cocci = nb_proies / nb_coccinelles

    # Quantité consommée selon l'abondance relative de proies
    if proies_par_cocci > 20:
        consomme = random.randint(12, 20)
    elif proies_par_cocci > 10:
        consomme = random.randint(8, 15)
    else:
        consomme = random.randint(3, 8)

    # On ne peut pas consommer plus que ce qui est disponible
    consomme = min(consomme, nb_proies)

    # Mise à jour du niveau de nutrition
    if consomme >= 10:
        self.niv_nutrition += 1
    else:
        self.niv_nutrition = max(0, self.niv_nutrition - 1)

    return nb_proies - consomme
```

Question 4 *Intégration de nouvelles règles biologiques*

```
def reproduction(self):
    """
    Une femelle avec un niveau de nutrition >= 2 et un âge >= 20 jours
    engendre exactement deux descendants : un mâle et une femelle.
    """
    descendants = []
    if self.sexe == "femelle" and self.niv_nutrition >= 2 and self.age >= 20:
        descendants.append(Coccinelle("male", 0, 0))
        descendants.append(Coccinelle("femelle", 0, 0))
        self.niv_nutrition = 0
    return descendants

def a_survecu(self):
    """
    Met à jour l'âge de la coccinelle et indique si elle est encore en vie.
    Une coccinelle meurt si elle a atteint son espérance de vie,
    ou si son niveau de nutrition est 0 et qu'un tirage aléatoire
    décide sa mort (probabilité 1/3).
    """
    self.age = self.age + 1
```

```
    if self.age >= self.esperance_de_vie:
        return False
    if self.niv_nutrition == 0 and random.random() < 1 / 3:
        return False
    return True
```

Test (résultat indicatif) :

```
>>> simulation_simple(population, 1000)   # age initial = 10 jours
(526, 0, 25)
```