Document d’accompagnement NSI

▼ Classe

▼ Monde

```
Monde pour une simulation

Attributs:
  _duree_repousse : Entier positif, Coefficient à partire duquel l'herbe aura poussée
  _dimension : Entier positif supérieur ou égal à 50, taille de la matrice
  _carte : Liste, matrice contenant tout les coefficients de pousse de l'herbe
```

```
Méthodes:
  __init__() :
    Constructeur de la classe Monde
          Données:
              _duree_repousse : Entier positif, Coefficient à partire duquel l'herbe aura poussée
              _dimension : Entier positif supérieur ou égal à 50, taille de la matrice
              _carte : Liste, matrice contenant tout les coefficients de pousse de l'herbe
          Résultat
              Crée un nouveau monde, ne renvoie rien

  herbe_pousse() :
    Fais pousser l'herbe de la carte
          Données: Aucuns paramètre pour cette méthode
          Résultat: Augmente chaque coefficient de la carte de 1, ne renvoie rien

  herbe_mangee() : Renvoie le status de l'herbe à une position donnée
    Renvoie le status de l'herbe à une position donnée
          Données:
              i : entier positif ou nul, abscisse dans la matrice
              j : entier positif ou nul, ordonnée dans la matrice
          Résultat:
              Renvoie un booléen, Vraie si l'herbe est en dessous de la durée de repousse
                                  Faux si l'herbe est au dessus de la durée de repousse

  nb_herbe() : Renvoie le nombre de carré d'herbe poussés
    Renvoie le nombre de carré d'herbe poussés
          Données: Aucuns paramètre pour cette méthode
          Résultat: Renvoie un entier positif, nombre de carré d'herbe poussés

  get_coef_carte() : Renvoi un coefficient de la matrice
    Renvoi un coefficient de la matrice
          Données:
              i : entier positif ou nul, abscisse dans la matrice
              j : entier positif ou nul, ordonnée dans la matrice
          Résultat:
              Renvoie un entier, coefficient de la matrice à une position donnée

  set_coef_carte() : Change un coefficient de la matrice
    Change un coefficient de la matrice
          Données:
              i : entier positif ou nul, abscisse dans la matrice
              j : entier positif ou nul, ordonnée dans la matrice
          Résultat:
              Change le coefficient de la matrice à une position donnée, ne renvoie rien

  get_dimension() : Renvoi les dimension de la carte
    Renvoi les dimension de la carte
          Données: Aucuns paramètre pour cette méthode
          Résultat: Renvoie un entier, dimension de la carte

  get_carte() : Renvoi la carte
    Renvoi la carte
```

```
          Données: Aucuns paramètre pour cette méthode
          Résultat: Renvoie une liste, matrice des coefficients de la carte
```

▼ Animaux

```
Animal pour une simulation

Attributs :
  _gain_nourriture : entier positif, energie gagnée en mangeant
  _position : tuple, abscisse et ordonnée de l'animal
  _energie : entier positif, energie dont dispose l'animal
```

```
Méthodes:
  __init__() :
    Constructeur de la classe Animaux
          Données:
              _gain_nourriture : entier positif, energie gagnée en mangeant
              _position : tuple, abscisse et ordonnée de l'animal
              _energie : entier, energie dont dispose l'animal
          Résultat:
              Crée un nouvel animal, ne renvoie rien

  deplacement() :
    Fais se déplacer l'animal
          Données:
              monde: Objet de type monde, matrice dans lequel se trouve l'animal
          Résultat:
              Déplace l'animal, ne renvoie rien
  get_gain_nourriture() :
    Renvoie combien d'énergie l'animal obtient en mangeant
          Données: Aucuns paramètre pour cette méthode
          Résultat: Renvoie un entier positif, énergie obtenue par l'animal en mangeant

  set_position() :
    Change la position de l'animal
          Données:
              i : entier positif ou nul, abscisse de l'animal
              j : entier positif ou nul, ordonnée de l'animal
          Résultat : Change la position de l'animal à un abscisse et une ordonnée donnés,
                      ne renvoie rien

  get_position() : Renvoie la position de l'animal
    Renvoie la position de l'animal
          Données: Aucuns paramètre pour cette méthode
          Résultat: Renvoie un tuple, position de l'animal

  set_energie() : Change l'énergie de l'animal
    Change l'énergie de l'animal
          Données:
              energie: Nombre entier, énergie que l'on veut mettre à l'animal
          Résultat : Change l'énergie de l'animal, ne renvoie rien

  get_energie() : Renvoie l'énergie de l'animal
    Renvoie l'énergie de l'animal
          Données: Aucuns paramètre pour cette méthode
          Résultat: Renvoie un entier, énergie dont dispose l'animal
```

▼ Mouton

```
Animal de type mouton, pour une simulation
Hérite de la classe Animaux
```

```
Méthodes:
  __init__():
```

```
    Constructeur de la classe Mouton
          Données:
              Attribut de la classe Animaux
          Résultat:
              Crée un nouveau mouton, ne renvoie rien

  variation_energie() :
    Fais varier l'énergie du mouton si
                      il est sur une case avec de l'herbe
          Données:
              monde: Objet de type Monde qui contient la matrice des coefficients de pousse de l'herbe
          Résultat:
              Enlève de l'énergie au mouton si il est sur une case d'herbe qui n'a pas poussée
              Lui en ajoute si il est sur une case avec de l'herbe poussée
              Renvoi l'energie qu'il lui reste
```

▼ Loup

```
Animal de type loup, pour une simulation
Hérite de la classe Animaux
```

```
Méthodes:
  __init__():
    Constructeur de la classe Loup
          Données:
              Attribut de la classe Animaux
              _cible : Mouton que le loup suit
          Résultat:
              Crée un nouveau loup, ne renvoie rien

  variation_energie() :
    Fais varier l'énergie du loup si
                      il est sur une case avec de l'herbe ou avec un mouton
          Données:
              simulation: Objet de type simulation qui contient la liste des moutons
          Résultat:
              Enlève de l'énergie au loup si il est sur de l'herbe,
              lui en ajoute si il est sur la même case qu'un mouton.
              Renvoi l'energie qu'il lui reste

  deplacement :
    Fais se déplacer le loup
          Données:
              monde: Objet de type monde, matrice dans lequel se trouve l'animal
              simulation : Objet de type simulation qui contient la liste des moutons
          Résultat:
              Déplace le loup vers un mouton si il y en a un dans un rayon
              aléatoire compris en 8 et 15 cases autour de lui puis le suit, sinon le déplace
              aléatoirement. Ne renvoie rien

  get_cible():
    Renvoie la cible du loup
          Données: Aucuns paramètre pour cette méthode
          Résultat: Renvoie un objet, cible que le loup suit
```

▼ Simulation

```
Classe qui gère la simulation

Attributs:
  _nombre_mouton() : Entier positif, nombre de mouton dans la simulation
  _horloge() : Entier positif ou nul, nombre de tour de la simulation
  _moutons() : Liste, contient tout les objets moutons de la simulation
  _loups() : Liste, contient tout les objets loups de la simulation
  _monde() : Classe de type Monde dans laquel vas tournée la simulation
  _resultats_herbe() : Entier positif, nombre de carrés d'herbes poussés dans la simulation
```

```
  _resultats_moutons() : Entier positif ou nul, nombre de moutons dans la simulation
  _resultats_loups() : Entier positif ou nul, nombre de loups dans la simulation
```

```
Méthodes:
  __init__() :
    Constructeur de la classe Simulation
          Données:
              _nombre_mouton() : Entier positif, nombre de mouton dans la simulation
              _horloge() : Entier positif ou nul, nombre de tour de la simulation
              _moutons() : Liste, contient tout les objets moutons de la simulation
              _loups() : Liste, contient tout les objets loups de la simulation
              _monde() : Classe de type Monde dans laquel vas tournée la simulation
              _resultats_herbe() : Entier positif,
                                   nombre de carrés d'herbes poussés dans la simulation
              _resultats_moutons() : Entier positif ou nul, nombre de moutons dans la simulation
              _resultats_loups() : Entier positif ou nul, nombre de loups dans la simulation
          Résultat:
              Crée une nouvelle simulation, ne renvoie rien

  simulation() :
    Fais tournée la simulation
          Données:
              monde: Objet de type monde dans laquel vas tournée la simulation
          Résultat:
              Augmente l'horloge
              Fait pousser l'herbe
              Change l'énergie des moutons et des loups, les tuant si ils n'en ont plus
              Fais se reproduire les moutons et les loups
              Compte le nombre de carrés d'herbe poussés, de mouton et de loup
              Ne renvoie rien

  get_mouton() :
    Renvoie la liste des moutons dans la simulation
          Données: Aucuns paramètre pour cette méthode
          Résultat: Renvoie une liste, contenant les moutons de la simulation

  nb_mouton() :
    Renvoie le nombre de mouton dans la simulation
          Données: Aucuns paramètre pour cette méthode
          Résultat: Renvoie un entier positif ou nul, nombre de moutons dans la simulation

  get_loup() :
    Renvoie la liste des loups dans la simulation
          Données: Aucuns paramètre pour cette méthode
          Résultat: Renvoie une liste, contenant les loups de la simulation

  nb_loup() :
    Renvoie le nombre de loup dans la simulation
          Données: Aucuns paramètre pour cette méthode
          Résultat: Renvoie un entier positif ou nul, nombre de loups dans la simulation
```

▼ Main

- Création du monde
- Création de la simulation
- Création d’une boucle qui s’arrête seulement si il n’y à plus de loup ou plus de mouton
  - Appel dans la boucle de la fonction simulation() qui vas exécuter toutes les actions qui doivent se faire à chaque boucles
  - Affichage l’interface pygame

▼ Répartition

▼ Arnaud

- Animaux
  - Loup
  - Mouton
- Assets
- PowerPoint

▼ Louis

- Simulation
- Interface
- Relecture et Github