■ Explication détaillée d’une fonction complexe utilisée dans le projet

■ Fonction : update_direction()

Cette fonction est au cœur du comportement autonome des entités dans la simulation.
Elle modélise une prise de décision directionnelle en fonction de l’environnement perçu :
nourriture, proies (entités de plus bas level) ou prédateurs (entités de plus haut level).

■ Objectif

- se diriger vers une cible (nourriture ou proie),
- fuir un danger (prédateur),
- conserver un comportement pseudo-aléatoire en l’absence de nourriture ou de danger.

■■ Principe de fonctionnement

1. Initialisation
- On initialise un vecteur d'attraction (attract_x, attract_y) à zéro.
- Un booléen attracted est utilisé pour détecter si un stimulus est présent.

2. Attraction vers la nourriture
- Pour chaque élément de nourriture visible :
  - On calcule le vecteur directionnel.
  - Le poids d’attraction est inversement proportionnel au carré de la distance,
    ce qui favorise les cibles proches.
  - Le vecteur résultant est ajouté au vecteur global d’attraction.

3. Attraction ou répulsion envers les entités
- Pour chaque entité visible :
  - On calcule la différence de niveau (self.level - entity.level) :
    - Si le niveau est positif → l’entité cible est une proie, donc attirance.
    - Si le niveau est négatif → l’entité cible est un prédateur, donc répulsion.
  - Le poids est proportionnel à la différence de niveau et toujours
    inversement proportionnel à la distance au carré.
  - Ce poids oriente le vecteur global d’attraction ou de répulsion.

4. Décision finale
- Si un ou plusieurs éléments ont influencé la direction (attracted == True)
  et que le vecteur n'est pas nul :
  - On calcule l’angle résultant du vecteur d’attraction par atan2.
- Sinon :
  - L’entité garde sa direction précédente mais y ajoute un bruit gaussien léger,
    simulant un comportement exploratoire aléatoire.

5. Normalisation de l’angle
- On applique angle % 360 pour s’assurer que la direction reste
  toujours comprise entre 0° et 360°.