W tym ćwiczeniu masz zaimplementować symulację łapania owcy przez wilka.

Symulacja obejmuje dwa rodzaje zwierząt: pojedynczego wilka i stado owiec. Zwierzęta poruszają się po łące o nieskończonych rozmiarach. Nie napotyka się tam żadnych przeszkód. Pomysł jest prosty - wilk próbuje polować na owce i jeść później, a owce uciekają - choć trochę niezdarnie.

Zgodnie z tym, co stwierdzono powyżej, łąka jest uważana za kartezjański układ współrzędnych - przestrzeń 2D ze środkiem w punkcie (0,0, 0,0).

Położenie każdego zwierzęcia na łące jest opisane przez dwie liczby zmiennoprzecinkowe - współrzędne. Mogą być zarówno pozytywne, jak i negatywne.

Na początku symulacji ustalana jest lokalizacja wszystkich zwierząt.

* 1. W przypadku owiec jest on losowo wybierany z równomiernego rozkładu z zakresu [-init\_pos\_limit, init\_pos\_limit].
  2. Dla wilka z kolei jest to środek łąki - punkt (0, 0, 0).

Symulacja przebiega w rundach według następującej kolejności:

* 1. Każda żywa owca porusza się:
     1. kierunek jest wybierany losowo (północ, południe, zachód lub wschód),
     2. Odległość ruchu w wybranym kierunku wynosi sheep\_move\_dist
  2. Wilk porusza się:
     1. Znajduje najbliższą owcę
     2. Sprawdza, czy jest to jego zasięg (odległość krótsza od wolf\_move\_dist).
        + - Jeśli tak, to zjada się owcę – owca znika, a wilk zajmuje jej miejsce
          - W przeciwnym razie wilk goni wybraną (najbliższą) owcę, czyli porusza się w jej kierunku wolf\_move\_dist
  3. Jeśli więcej niż jedna owca znajduje się w zasięgu wilka, ścigana jest tylko jedna.

Symulacja kończy się, gdy tylko wszystkie owce zostaną zjedzone lub zostanie osiągnięta maksymalna liczba rund.

**Szczegółowe wymagania dla grupy zaszeregowania 3:**

1. Aby zrealizować symulację zgodną z powyższym opisem, przyjmując następujące wartości parametrów:
   * maksymalna liczba rund: 50;
   * wielkość stada owiec: 15;
   * init\_pos\_limit: 10,0;
   * sheep\_move\_dist: 0,5;
   * wolf\_move\_dist: 1.0.

Aby uzyskać liczby losowe, użyj wbudowanego pakietu *losowego* lub *NumPy*.

1. Do implementacji wyświetlanie podstawowych informacji o stanie symulacji tuż po zakończeniu rundy. Należy wziąć pod uwagę następujące informacje:
   * numer rundy;
   * pozycja wilka (z dokładnością do 3 miejsca po przecinku);
   * liczbę żywych owiec;
   * jeśli wilk goni jakąkolwiek owcę – która to jest (co jest jej identyfikatorem lub numerem porządkowym używanym podczas inicjalizacji);
   * Jeśli jakaś owca została zjedzona - wyświetl informację, która z nich.

Wyświetlenie powyższych informacji nie powinno przerywać symulacji. Nie zakłada się interakcji z użytkownikiem.

1. Korzystając z pakietu json, zaimplementuj zapisywanie lokalizacji każdego zwierzęcia dla każdej rundy, do pliku pos.json. Treść powinna być listą słowników, z których każda odpowiada pojedynczej rundzie symulacji i zawiera następujące dane:
   * "round\_no" - liczba rundy (liczba całkowita);
   * "wolf\_pos" - położenie wilka (para liczb zmiennoprzecinkowych);
   * "sheep\_pos" - lokalizacja wszystkich owiec (lista zawierająca parę liczb zmiennoprzecinkowych dla każdej żywej owcy lub None/null dla każdej zjedzonej owcy).

Plik JSON powinien być dobrze sformatowany. Należy dodać odpowiednie wcięcie. Jeśli plik pos.json już istnieje, należy go zastąpić.

1. Korzystając z pakietu *csv*, zaimplementuj zapisywanie liczby żywych owiec w każdej rundzie do pliku alive.csv. Powinien mieć dwie kolumny:
   * okrągła liczba (wartość całkowita);
   * liczba żywych owiec (wartość całkowita).

Każdy wiersz w pliku powinien odpowiadać pojedynczej rundzie. Jeśli plik już istnieje.csv — nadpisz go.

**Szczegółowe wymagania dla klasy 4:**

1. Wszystkie wymagania dla niższej klasy powinny być spełnione.
2. Użyj pakietu *argparse* do obsługi argumentów wiersza polecenia:
   * -c/--config FILE — pomocniczy plik konfiguracyjny, gdzie FILE określa nazwę pliku;
   * -d/--dir DIR — podkatalog, w którym powinny być umieszczone pliki pos.json, alive.csv i — opcjonalnie — chase.log, gdzie DIR oznacza nazwę podkatalogu;
   * -h/--help — pomoc;
   * -l/--log LEVEL — jeśli zdarzenia powinny być rejestrowane, gdzie LEVEL jest poziomem rejestrowania (*DEBUG*, *INFO*, *WARNING*, *ERROR* lub *CRITICAL*);
   * -r/--rounds NUM — liczba rund, gdzie NUM jest liczbą całkowitą;
   * -s/--sheep NUM — liczba owiec w stadzie, gdzie NUM jest wartością całkowitą;
   * -w/--wait — jeśli symulacja powinna być wstrzymywana na koniec każdej rundy po wyświetleniu podstawowych informacji o stanie symulacji; aby kontynuować, każdy powinien zostać naciśnięty przez użytkownika.

Wszystkie argumenty powinny być opcjonalne i powinny być przetwarzane zgodnie z poniższymi zasadami.

* + Opcja -c/--config wskazuje plik konfiguracyjny, w którym zapisywane są informacje o init\_pos\_limit, sheep\_move\_dist i wolf\_move\_dist.  Format jest opisany poniżej. Jeśli plik konfiguracyjny nie został przekazany, należy użyć wartości domyślnych dla init\_pos\_limit, sheep\_move\_dist i wolf\_move\_dist (zgodnie z wymaganiami dla stopnia 3).
  + Opcja -d/--dir wskazuje podkatalog do zapisywania plików wyjściowych. Jeśli żądany katalog nie istnieje, należy go utworzyć. Jeśli opcja -d/--dir nie została przekazana, wszystkie pliki powinny być przechowywane w bieżącym katalogu.
  + Opcja -h/--help pokazuje podręcznik programu. Powinien zawierać krótki opis symulacji i informacje o jego parametrach. Użycie tej opcji powoduje zamknięcie symulacji tuż po wyświetleniu instrukcji — symulacja nie jest uruchamiana.
  + Opcja -l/--log definiuje poziom rejestrowania. Jeśli ta opcja zostanie przekazana, w bieżącym katalogu lub podkatalogu wskazanym przez parametr -d/--dir zostanie utworzony pościg za plikiem.log . Plik powinien zostać zastąpiony, jeśli istnieje. Jeśli opcja -l/--log nie zostanie przekazana, plik chase.log nie zostanie utworzony i nie zostaną zarejestrowane żadne zdarzenia.
  + Opcja -r/--rounds to maksymalna liczba rund w symulacji.
  + Opcja -s/--owca wskazuje liczbę owiec w stadzie.
  + Opcja -w/--wait jest flagą, która wskazuje, że symulacja powinna być wstrzymywana po każdej rundzie, dopóki użytkownik nie naciśnie dowolnego. Jeśli ta opcja nie zostanie zaliczona, symulacja nie powinna zostać przerwana.

Wszystkie argumenty powinny być odpowiednio zweryfikowane (np. liczba rund jest liczbą całkowitą większą od zera). Jeśli weryfikacja nie powiedzie się, użytkownik powinien zostać poinformowany o błędzie za pomocą odpowiedniego wyjątku.

1. Za pomocą pakietu *configparser* zaimplementuj wartości ładowania dla init\_pos\_limit, sheep\_move\_dist i wolf\_move\_dist z pliku konfiguracyjnego zdefiniowanego w argumencie -c/--config. Plik konfiguracyjny powinien być w formacie INI zgodnie z poniższą strukturą:
2. [Pole]
3. InitPosLimit = 10,0
4. [Ruch]
5. SheepMoveDist = 0.5

WolfMoveDist = 1.0

Wszystkie wartości powinny zostać zatwierdzone (np. długość, o którą poruszają się zwierzęta, jest dodatnia). W przypadku niepowodzenia procesu walidacji użytkownik powinien zostać o tym poinformowany w drodze odpowiedniego wyjątku.

1. Korzystając z pakietu *rejestrowania*, zaimplementuj zdarzenia rejestrowania w pliku o nazwie chase.log. Wszystkie zdarzenia na poziomie logowania określonym przez -l/--log lub wyższym powinny być rejestrowane:
   * dla poziomu *DEBUG* (10) należy rejestrować wszystkie wywołania funkcji, a także wartości zwracane przez te funkcje (jeśli istnieją);
   * dla poziomu *INFO* (20) wszystkie czynności podejmowane przez program powinny być rejestrowane (np. ustalanie początkowych pozycji zwierząt, ruch zwierząt itp.);
   * dla poziomów *WARNING* (30), *ERROR* (40) i *CRITICAL* (50) należy zarejestrować odpowiednie zdarzenia.

Poziom rejestrowania *DEBUG* jest przeznaczony dla deweloperów do śledzenia możliwych błędów w programie. Z kolei poziom *INFO* jest przeznaczony dla użytkowników i ma na celu przechowywanie informacji o przebiegu symulacji, aby można ją było zrekonstruować w przyszłości. W przypadku wyższych poziomów, takich jak *WARNING*, *ERROR* i *CRITICAL,* rejestrowane zdarzenia zależą od sposobu implementacji programu. Jeśli żadne zdarzenia nie wydają się odpowiednie do rejestrowania na tych poziomach, nie jest wymagane rejestrowanie czegokolwiek.

**Szczegółowe wymagania dla grupy zaszeregowania 5:**

1. Wszystkie wymagania dla niższych klas powinny być spełnione.
2. Użyj pakietu *distutils* lub *setuptools*, aby utworzyć publikowalny pakiet *chase —* instalowalny w środowisku wirtualnym. Kod powinien być dobrze zorganizowany, aby logika mogła być ponownie użyta. Fragment kodu odpowiedzialny za symulację przebieganą w trybie konsolowym powinien znajdować się w module \_\_main\_\_.py wewnątrz pakietu. Po zainstalowaniu pakietu zostanie uruchomiona symulacja z poleceniem python -m chase [ARG], gdzie ARG są opcjonalnymi argumentami symulacji (patrz: wymagania dla klasy 4).
3. Podczas obrony projektu powinieneś pokazać, jak stworzyć środowisko wirtualne i jak zainstalować nowo utworzony pakiet *chase.*  Następnie należy uruchomić symulację z zainstalowanego pakietu