# Python. Домашнее задание 1

Преподаватели: Дмитрий Косицин, Светлана Боярович и Анастасия Мицкевич

**Задание 1. Игра "Жизнь"**

Напишите программу, моделирующую экологическую систему океана, в котором обитают хищники и жертвы (добыча). Океан представляется двухмерным массивом ячеек. В ячейке может находиться или хищник, или добыча, или препятствие. В каждый квант времени ячейки последовательно обрабатываются. Хищник может съесть соседнюю добычу или просто переместиться на свободную клетку. Добыча также может переместиться на свободную клетку. Если в течение определенного времени хищник ничего не съел, то он погибает. Через определенные интервалы времени добыча и хищники размножаются (если рядом есть совбодная клетка). При этом потомок занимает эту свободную клетку.

Начальные параметры задаются в файле. Моделирование заканчивается либо когда пройдет определенное число итераций, либо когда погибнут все хищники или вся добыча.

Карта океана (положение всех объектов) должно задаваться также в файле. Предусмотрите возможность сгенерировать карту океана случайным образом (можно в дополнительном скрипте).

При написании решения разрешено пользоваться только стандартной библиотекой. Старайтесь соблюдать code style, писать функционально и не использовать глобавльные переменные для хранения состояния. Использовать классы разрешено.

Ради интереса понаблюдайте, как ведет себя популяция в зависимости от начальных значений. Свои наблюдения можете описать в файле и прикрепить к сделанной задаче.

Программа должна находиться в файле model.py и принимать следующие аргументы командной строки:

- -i – количество итераций
- -c – файл с общей конфигурацией (время жизни существ, периоды размножения и т.д.). Формат можете выбрать произвольно, но рекомендуется использовать json. К решению приложите один файл с работающей конфигурацией.
- -o – файл, в который записаны результаты (карта океана) моделирования. Формат может быть произвольным.

Пример: model.py -i 100 -c config.json -o result.txt

Напишите также юнит-тесты к программе и сохраните их в файле model_ut.py. Тесты должны либо быть набором функций (с префиксом *test_* ), содержащими assert'ы и тестирующими поведение определенных функций внутри вашей программы, либо test case'ами, использующими библиотеку unittest (будет проходить на следующей паре).

**Задание 2.** Напишите код интерпретатора арифметических выражений. Он должен поддерживать работу с операциями $+, -, *, /, **$ и скобками (любой вложенности) для целых и дробных чисел. Числа могут быть отрицательными. Унарный минус для скобок поддерживать не нужно. Функцией eval пользоваться нельзя. Приоритет операторов такой как в Python. Деление целочисленным не является.

На вход функции с названием interpret подается *корректное* арифметическое выражение, содержащее до 250 элементов разделенных пробелами. Возможные элементы: $+, -, *, /, **, (, )$, целые и дробные числа. Минус с отрицательными числами пишется слитно. Функция должна возвращать результат вычисления выражения.

Сама программа должна содержать указанную выше функцию и работать как интерпретатор, считывая в цикле строки с консоли и выводя результат на консоль.

***Пример***. Для входа '$2 * (3 + 5)$' результат – '16'.

***Пример***. Для входа '$1 + 2 * (1 + 2 * (1 + 2 * (-1 + 2)))$' результат – '15'.