**Задание 7**

**Задачи для седьмого задания Практикума по программированию. Общая тема задания «Реализация собственного пакета модулей для манипулирования плоскими фигурами».**

Базовые требования к функциональности программы сохраняются прежними:

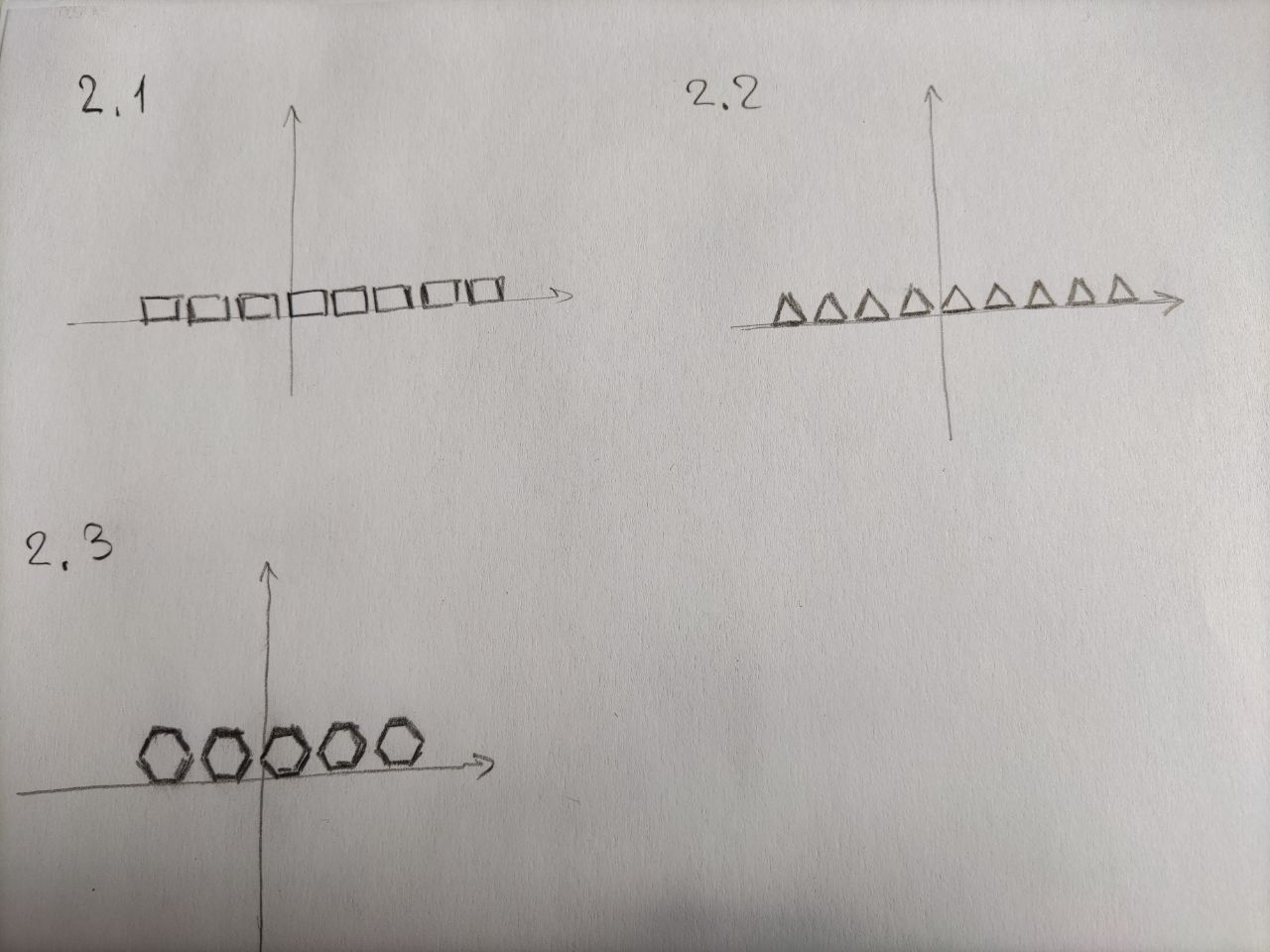
Реализовать api, которое позволяет: генерировать, преобразовывать и визуализировать последовательность плоских полигонов, представленных в виде кортежа кортежей (например: ((0,0), (0,1), (1,1), (1,0)) – представление для квадрата). Последовательности представлений полигонов представляют из себя итераторы (далее: последовательности полигонов). Решать задачи с использованием функционального стиля программирования, в том числе активно использовать функции из модуля itertools и functools.

1. Реализовать функцию визуализации последовательности полигонов, представленной в виде итератора (например, можно использовать визуализацию с помощью библиотеки matplotlib, см. пример): <https://matplotlib.org/stable/gallery/shapes_and_collections/patch_collection.html#sphx-glr-gallery-shapes-and-collections-patch-collection-py>

***(обязательная часть)***

1. Реализовать функции, генерирующие бесконечную последовательность не пересекающихся полигонов с различающимися координатами (например, «ленту»):
   1. прямоугольников ( **gen\_rectangle()** );
   2. треугольников ( **gen\_triangle()** );
   3. правильных шестиугольников ( **gen\_hexagon()** ).
   4. с помощью данных функций используя функции из модуля itertools сгенерировать 7 фигур, включающих как прямоугольники, так и треугольники и шестиугольники, визуализировать результат.

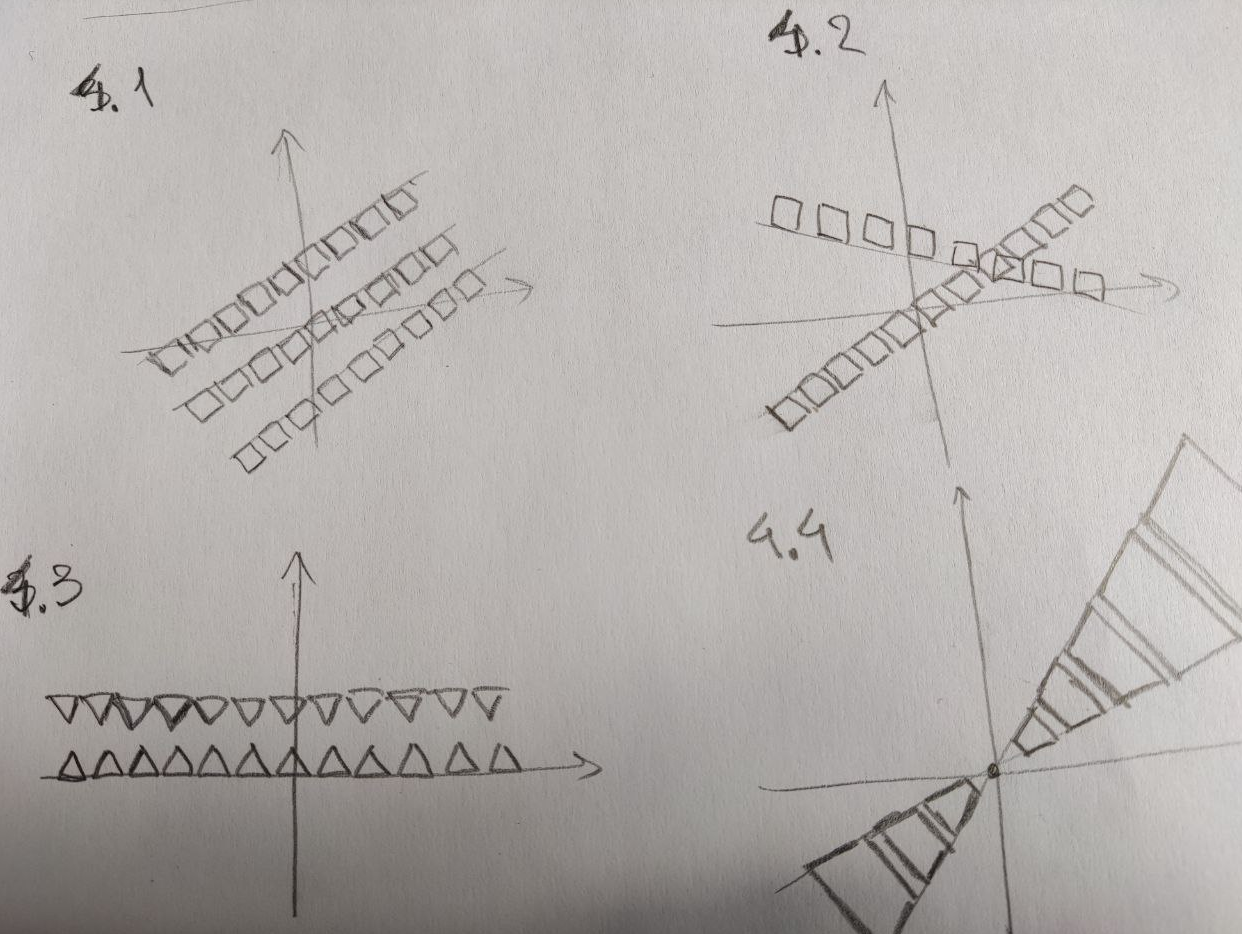
***(обязательная часть)***



1. Реализовать операции:
   1. параллельный перенос ( **tr\_translate** );
   2. поворот ( **tr\_rotate** );
   3. симметрия ( **tr\_symmetry** );
   4. гомотетия ( **tr\_homothety** );

которые можно применить к последовательности полигонов с помощью функции map.

***(обязательная часть)***



1. С помощью данных функций создать и визуализировать:
   1. 3 параллельных «ленты» из последовательностей полигонов, расположенных под острым углом к оси x;
   2. две пересекающихся «ленты» из последовательностей полигонов, пересекающихся не в начале координат;
   3. две параллельных ленты треугольников, ориентированных симметрично друг к другу;
   4. последовательность четырехугольников в разном масштабе, ограниченных двумя прямыми, пересекающимися в начале координат (см. рис.).

***(обязательная часть)***

1. Реализовать операции:
   1. фильтрации фигур, являющихся выпуклыми многоугольниками ( **flt\_convex\_polygon** );
   2. фильтрации фигур, имеющих хотя бы один угол, совпадающий с заданной точкой ( **flt\_angle\_point** );
   3. фильтрации фигур, имеющих площадь менее заданной ( **flt\_square** );
   4. фильтрации фигур, имеющих кратчайшую сторону менее заданного значения ( **flt\_short\_side** );
   5. фильтрации выпуклых многоугольников, включающих заданную точку (внутри многоугольника) ( **flt\_point\_inside** );
   6. фильтрации выпуклых многоугольников, включающих любой из углов заданного многоугольника ( **flt\_polygon\_angles\_inside** );

которые можно применить к последовательности полигонов с помощью функции filter.

***(обязательная часть: 2 пункта, 4 пункта – сложность 1; 6 пунктов – сложность 2)***

1. С помощью данных функций реализовать и визуализировать:
   1. фильтрацию фигур, созданных в рамках пункта 4.4; подобрать параметры так, чтобы на выходе было получено 6 фигур;
   2. используя функции генерации из п. 2 и операции из п. 3 создать не менее 15 фигур, которые имеют различный масштаб и выбрать из них (подбором параметра фильтрации) не более 4х фигур, имеющих кратчайшую сторону менее заданного значения;
   3. используя функции генерации из п. 2 и операции из п. 3 создать не менее 15 фигур имеющих множество пересечений и обеспечить фильтрацию пересекающихся фигур.

***(обязательная часть: 1 пункт, 3 пункта – сложность 1)***

1. Реализовать декораторы и продемонстрировать корректность их работы:
   1. Фильтрующие многоугольники в итераторах среди аргументов функции, работающие на основе функций из 5: **@flt\_convex\_polygon, @flt\_angle\_point, @flt\_square, @flt\_short\_side, @flt\_point\_inside, @flt\_polygon\_angles\_inside** ;
   2. Преобразующие многоугольники в итераторах среди аргументов функции, работающие на основе функций из 3: **@tr\_translate, @tr\_rotate, @tr\_symmetry, @tr\_homothety** ;

***(обязательная часть: 1 пункта, 5 пунктов – сложность 1)***

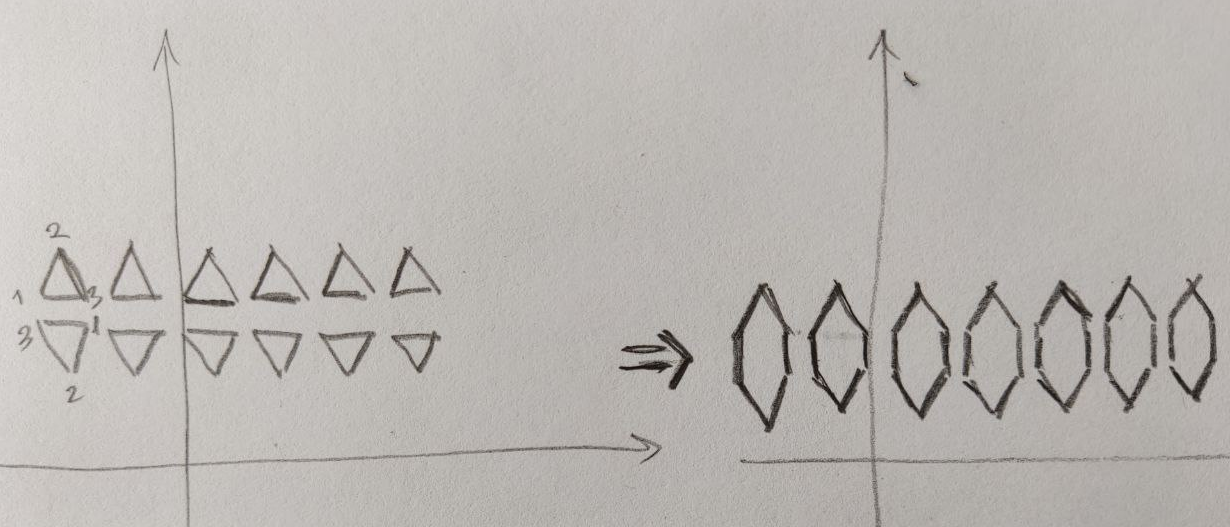
1. Реализовать функции и продемонстрировать их корректность:
   1. поиск угла, самого близкого к началу координат ( **agr\_origin\_nearest** );
   2. поиск самого длинной стороны многоугольника ( **agr\_max\_side** );
   3. поиск самой маленькой площади многоугольника ( **agr\_min\_area** );
   4. расчет суммарного периметра ( **agr\_perimeter** );
   5. расчет суммарной площади ( **agr\_area** ).

которые можно применить к последовательности полигонов с помощью функции [functools](https://docs.python.org/3/library/functools.html#module-functools).reduce .

***(3 пункта – сложность 1; 5 пунктов – сложность 2)***

1. Реализовать функции и продемонстрировать пример их работы (если возможно, с визуализацией):
   1. склейки полигонов в одну последовательность полигонов из нескольких последовательностей полигонов [**zip\_polygons(iterator1, iterator2, [iterator3, …])**](https://docs.python.org/3/library/itertools.html#itertools.accumulate). Пример:  [**zip\_polygons([((1,1), (2,2), (3,1)), ((11,11), (12,12), (13,11))], [((1,-1), (2,-2), (3,-1)), ((11,-11), (12,-12), (13,-11))])**](https://docs.python.org/3/library/itertools.html#itertools.accumulate) ->[((1,1), (2,2), (3,1), (1,-1), (2,-2), (3,-1)), ((11,11), (12,12), (13,11), (11,-11), (12,-12), (13,-11))] .

Альтернативный пример (визуализация):



* 1. генерации **count\_2D()** параметры: (start1, start2), [(step1, step2)], результаты: (start1, start2), (start1+step1, start2+step2), (start1+2\*step1, start2+2\*step2)
  2. склейки полигонов в одну последовательность полигонов из нескольких последовательностей **zip\_tuple(iterator1, iterator2) . Пример: zip\_tuple([(1,1), (2,2), (3,3), (4,4)], [(2,2), (3,3), (4,4), (5,5)], [(3,3), (4,4), (5,5), (6,6)]) ->** ((1,1), (2,2), (3,3)), ((2,2), (3,3) (4,4)), ((3,3), (4,4), (5,5)), ((5,5), (6,6), (7,7))

***(3 пункта – сложность 1)***

**Для каждого студента формируется комплексное задание из сочетания пунктов. Суммарная сложность комплексного задания должна быть не менее 5, как минимум одна задача из комплекта должна стоить дороже 1. Приветствуется выполнение заданий с суммарной сложностью более 5 (рекомендуется отмечать такие решения дополнительными баллами).**

**Сложность комплексного задания может быть скорректирована преподавателем в зависимости от сочетания пунктов между собой. В случае неоднозначности сочетания требований двух пунктов студент предлагает преподавателю свое видение итоговой постановки задачи, а преподаватель согласует его (рекомендуется фиксировать договоренность в письменном виде). Преподаватель имеет право увеличить оценку сложности комплексного задания в случае существенного увеличения трудоемкости при сочетании двух пунктов.**

**Преподаватель имеет право добавить пункты с собственной постановкой задачи и оценкой сложности. Студент может предложить свои пункты и реализовывать их в случае предварительно согласования с преподавателем постановки задачи и оценки сложности пункта. Рекомендуется фиксировать дополнительные пункты письменно. Желательно, чтобы дополнительные пункты составляли не более половины суммарной трудоемкости комплексного задания.**