

Современные методы программирования

Судаков Владимир Анатольевич

sudakov@ws-dss.com

2022

План

- Введение/Знакомство
- Python
 - Динамические аспекты и ООП
 - Объектная модель
 - Встроенные типы
 - Функции

NumPy

Pandas

ИСТОЧНИКИ

- Марк Лутц. Изучаем Python. 5-е изд.
- Luciano Ramalho. Fluent Python
- Joel Grus. Data Science from Scratch
- Allen B. Downey. Think Complexity
- PEP8
- Matloff, Norman S. The art of R programming: tour of statistical software design
- https://github.com/sudakov/lab_it

Языки программирования



Эскимосы и снег

- Что этот язык может «делать»?
- Программа – последовательность символов, определяющая вычисления
- Язык программирования – это набор правил, определяющих, какие последовательности символов составляют программу и какое вычисление описывает программа

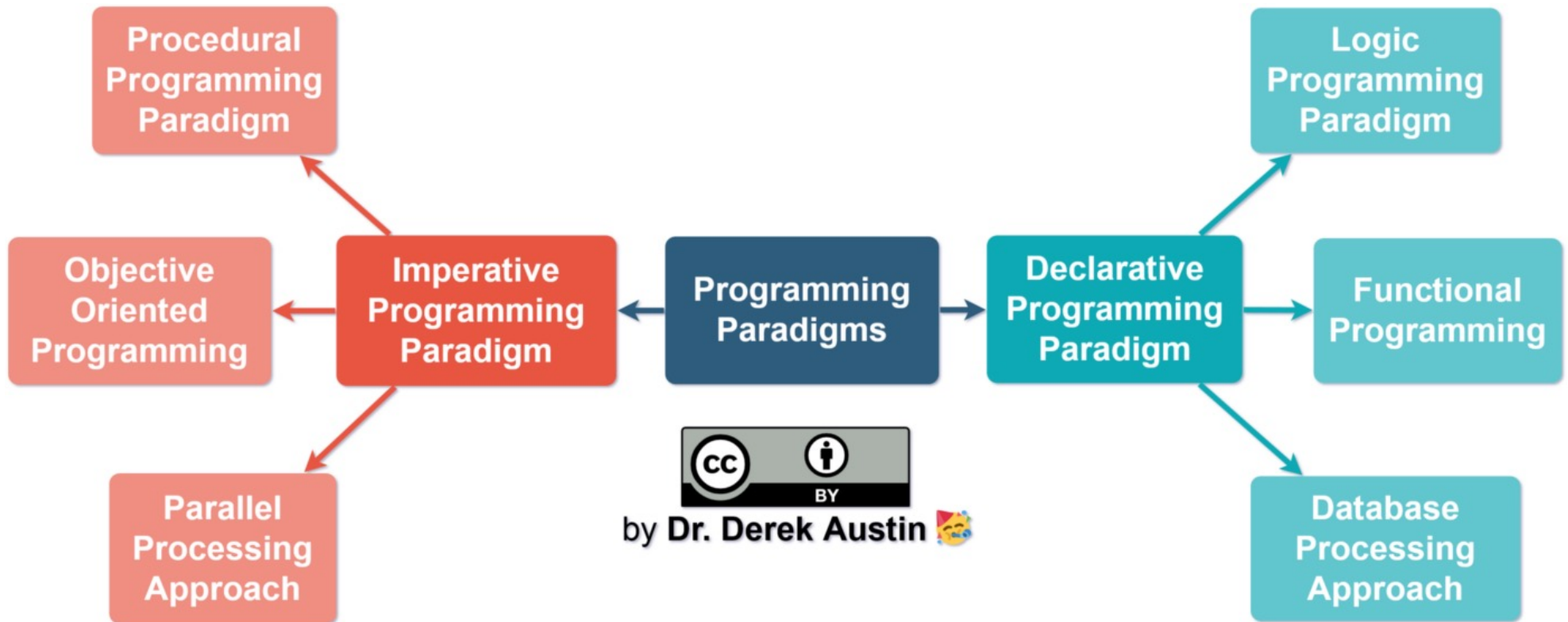
Что нужно знать

- Python или R или
- SQL
- XML, JSON, HTML
- Git
- Парадигмы

Оно нам надо?

- Давайте посмотрим hh
- Должен ли Data Scientist видеть базу?
- Что мы будем делать в будущем?

Парадигмы



Давайте решим задачу

- Дан список списков: $[[1,2,3,\dots],[4,5,7,\dots],\dots]$
- Найти сумму вторых элементов всех вложенных списков: $2+5+\dots$
- Предложите разные решения на Python
- Может быть есть красивые решения на других языках?
- Сколько человек выбрали, то или иное решение?
- Какое решение лучше и почему?

Python

- Интерпретация
- Байт-код
- Встроенные типы реализованы на C
- Динамический
- Мультипарадигменный
- Эталонной реализацией Python является интерпретатор CPython. Распространяется под свободной лицензией Python Software Foundation License. Есть реализация для JVM с возможностью компиляции, CLR. PyPy использует JIT-компиляцию, которая значительно увеличивает скорость выполнения Python-программ.

Где писать код?

- Jupyter
- <https://colab.research.google.com>
- Текстовый редактор
- Atom
- PyCharm

А где выполнять?

- консоль
- python.org
- см. выше

PEP8

Yes:

```
i = i + 1  
submitted += 1  
x = x*2 - 1  
hypot2 = x*x + y*y  
c = (a+b) * (a-b)
```

No:

```
i=i+1  
submitted +=1  
x = x * 2 - 1  
hypot2 = x * x + y * y  
c = (a + b) * (a - b)
```

PEP8

- Используйте пробелы, а не табуляции
- Строка не более 79 символов

```
with open('/path/to/some/file/you/want/to/read') as file_1, \
    open('/path/to/some/file/being/written', 'w') as file_2:
    file_2.write(file_1.read())
```

import this

- Красивое лучше, чем уродливое.
- Явное лучше, чем неявное.
- Простое лучше, чем сложное.
- Сложное лучше, чем запутанное.
- Плоское лучше, чем вложенное.
- Разреженное лучше, чем плотное.
- Читаемость имеет значение.
- Особые случаи не настолько особые, чтобы нарушать правила.
- При этом практичность важнее безупречности.
- Ошибки никогда не должны замалчиваться.
- Если не замалчиваются явно.
- Встретив двусмысленность, отбрось искушение угадать.
- Должен существовать один — и, желательно, *только* один — очевидный способ сделать это.
- Хотя он поначалу может быть и не очевиден, если вы не голландец.
- Сейчас лучше, чем никогда.
- Хотя никогда зачастую лучше, чем *прямо* сейчас.
- Если реализацию сложно объяснить — идея плоха.
- Если реализацию легко объяснить — идея, *возможно*, хороша.
- Пространства имён — отличная вещь! Давайте будем делать их больше!

ООП

Объектно-ориентированное программирование (сокр. ООП) — методология программирования, основанная на представлении программы в виде совокупности взаимодействующих объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

Объект

Объект в программировании — сущность в цифровом пространстве, обладающая состоянием и поведением, имеющая поля и методы. Как правило, при рассмотрении объектов выделяется то, что объекты принадлежат одному или нескольким классам, которые определяют поведение (являются моделью) объекта. Термины «экземпляр класса» и «объект» взаимозаменяемы. Аналогия: объект - то что сделали по чертежу, класс - это чертёж.

Класс

Класс — в объектно-ориентированном программировании, модель для создания объектов определённого типа, описывающая их структуру (набор полей и их начальное состояние) и определяющая алгоритмы (функции или методы) для работы с этими объектами

Основные принципы ООП

- абстракция для выделения в моделируемом предмете важного для решения конкретной задачи по предмету, в конечном счёте — контекстное понимание предмета, формализуемое в виде класса;
- инкапсуляция для быстрой и безопасной организации собственно иерархической управляемости: чтобы было достаточно простой команды «что делать», без одновременного уточнения как именно делать, так как это уже другой уровень управления;
- наследование для быстрой и безопасной организации родственных понятий: чтобы было достаточно на каждом иерархическом шаге учитывать только изменения, не дублируя всё остальное, учтённое на предыдущих шагах;
- Полиморфизм - способность метода обрабатывать данные разных типов.

Метод k ближайших соседей

- Метрика близости
- Голосование k ближайших соседей
- Если результат равный – то убираем самого дальнего соседа или считаем средневзвешенный голос
- Для целей обучения именно программированию не используем **Scikit-learn** или аналоги, хотя можно использовать для сравнения...

Задача

- Давайте познакомимся:
 - Какая у Вас ближайшая станция метро?
 - Что Вы пьете по утрам? Чай или Кофе?
- Разбиваемся на команды 3-4 человека:
 - Распределение ролей
 - Парное программирование
 - Подготовка исходных данных
 - Тестирование
 - Анализ – какое к лучше?
 - Показ решения
- Обсуждение
 - Какое решение лучше и почему?

Модель данных Python

Jupyter

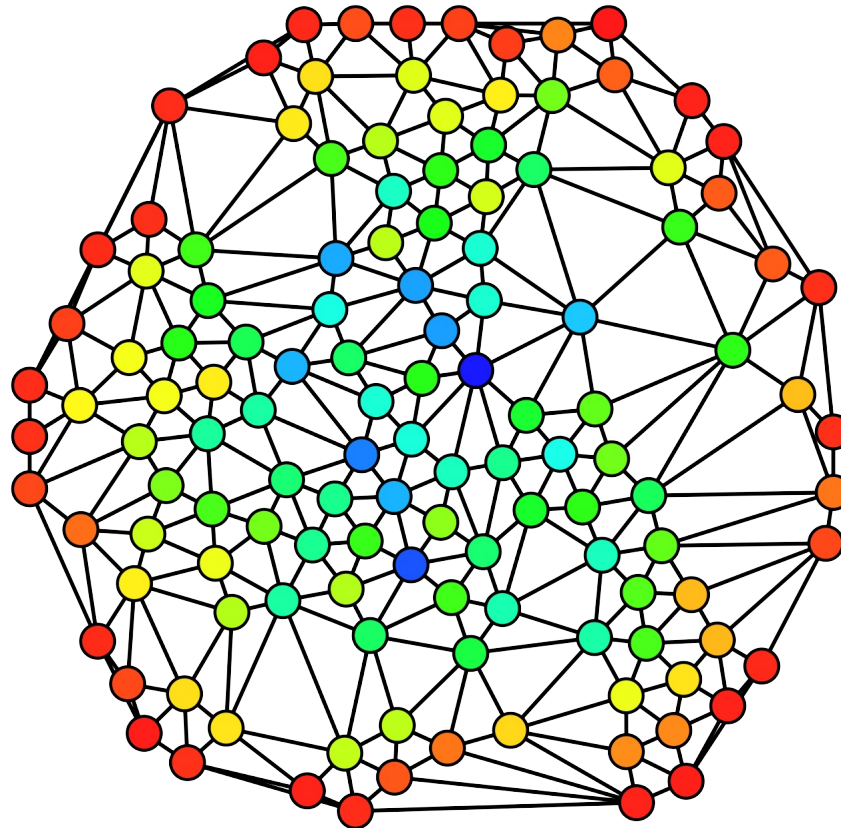
Переменные

- Переменные - это не ящики, это этикетки

Jupyter....

Задача

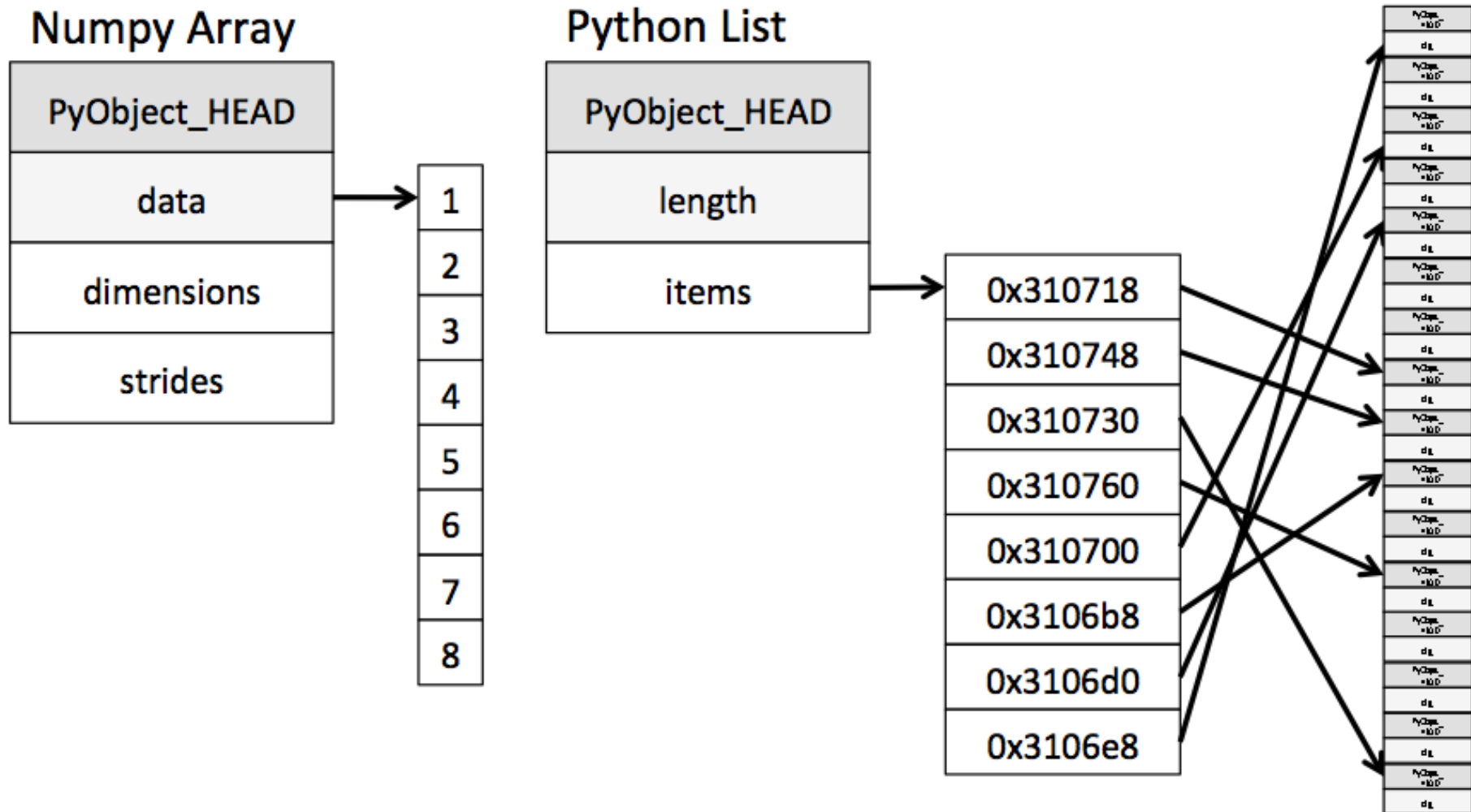
- Давайте соберем информацию о друзьях из VK
- Оценить центральность: по посредничеству, по близости, собственного вектора



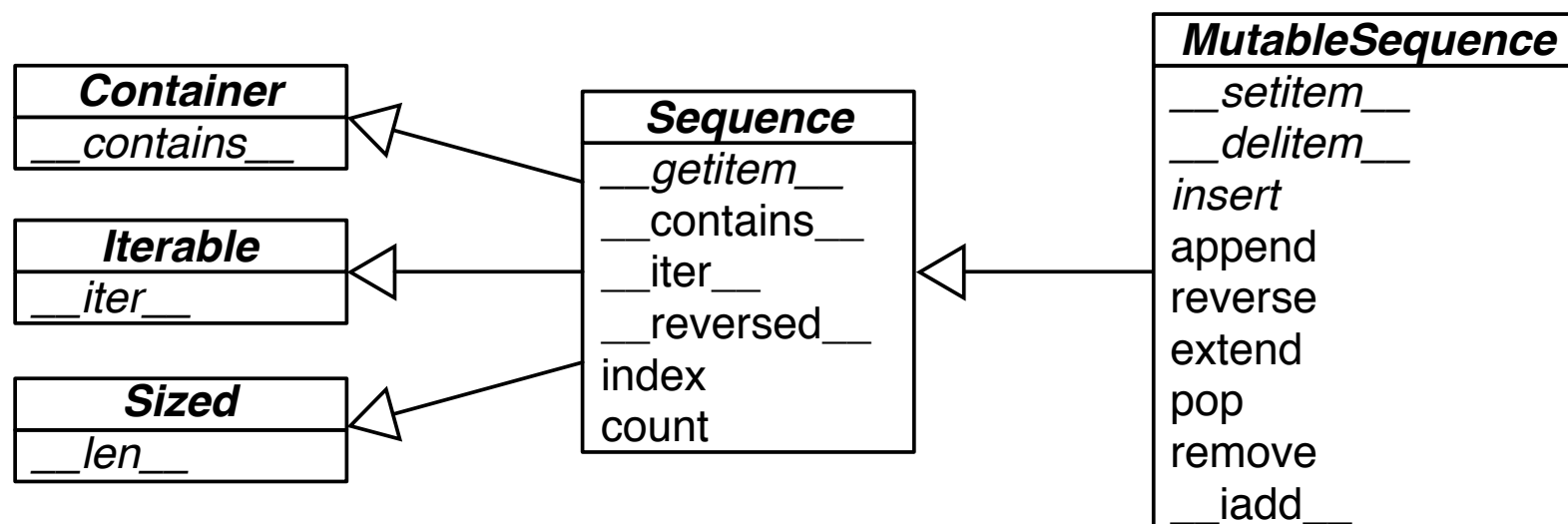
Виды последовательностей

- Контейнерные:
 - list, tuple, collections.deque
- Плоские:
 - str, bytes, bytearray, memoryview, array.array
- Изменяемые:
 - list, bytearray, collections.deque, memoryview, array.array
- Неизменяемые:
 - tuple, str, bytes

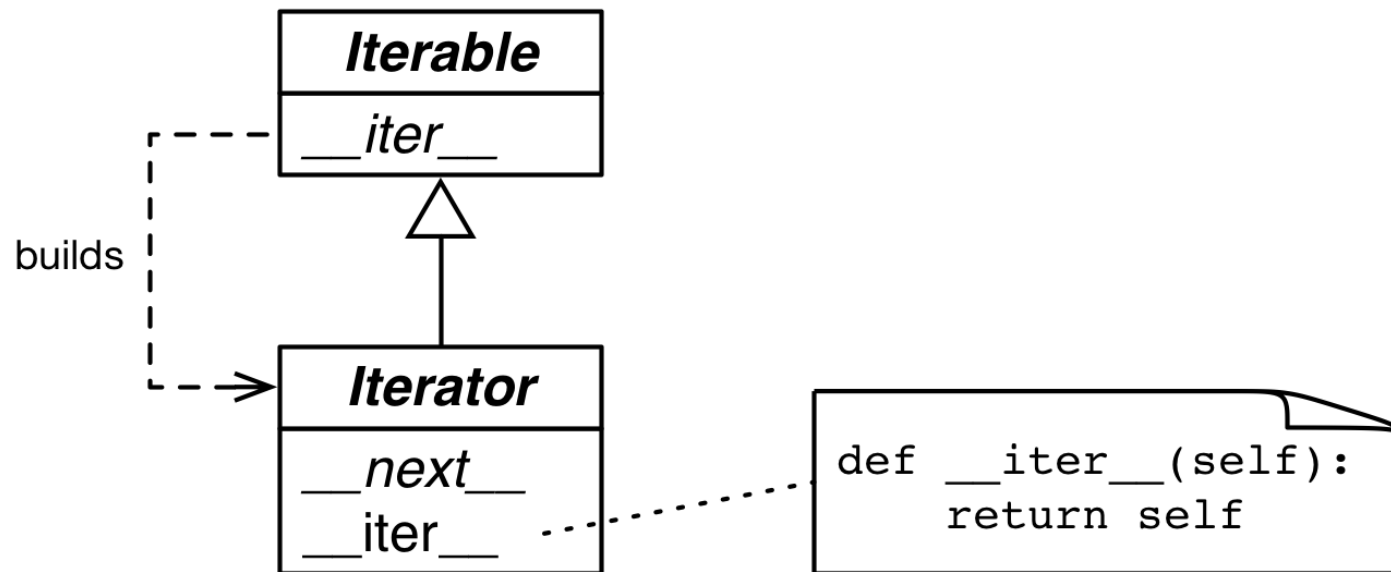
Представление в памяти



Последовательности



Итератор



Итератор

- Решает проблему просмотра данных не помещающихся в память
- Он делает это лениво....
- Итераторы используются для поддержки:
 - for
 - Конструирования коллекций
 - Построчного просмотра файлов
 - Списковых и словарных включений
 - Распаковки кортежей
 - Распаковки фактических параметров *

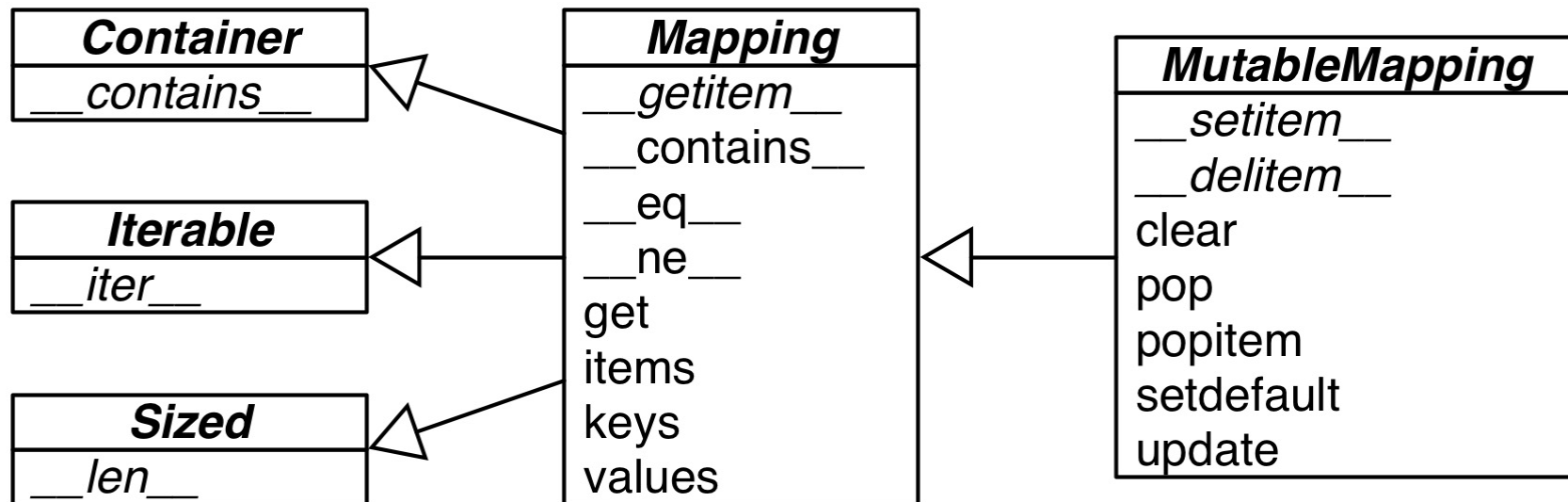
Домашнее задание

- Реализовать модель Шеллинга (модель расовой сегрегации)
- Дан квадрат $n \times n$. 45% клеток синие, 45% клеток красные, 10% клеток пустые. Начальное заполнение в случайном порядке.
- Клетка «счастлива» если у нее 2 или более соседа одного с ней цвета. Соседи – это 8 клеток вокруг данной.
- Моделирование: выбрать случайным образом «несчастную» клетку и переместить ее в случайно выбранную пустую клетку.
- Вывести квадраты через данное некоторое количество шагов иллюстрирующее расовую сегрегацию.

Домашнее задание

1. Дано n отрезков на плоскости. Определить: Есть ли среди них пересечения?
2. Дано n точек на плоскости. Построить выпуклый многоугольник, который включает все эти точки.
3. На плоскости дан многоугольник (необязательно выпуклый). Дана точка. Определить принадлежит ли точка многоугольнику.
4. Дано n точек на плоскости. Найди окружность минимального радиуса которой принадлежат все эти точки.

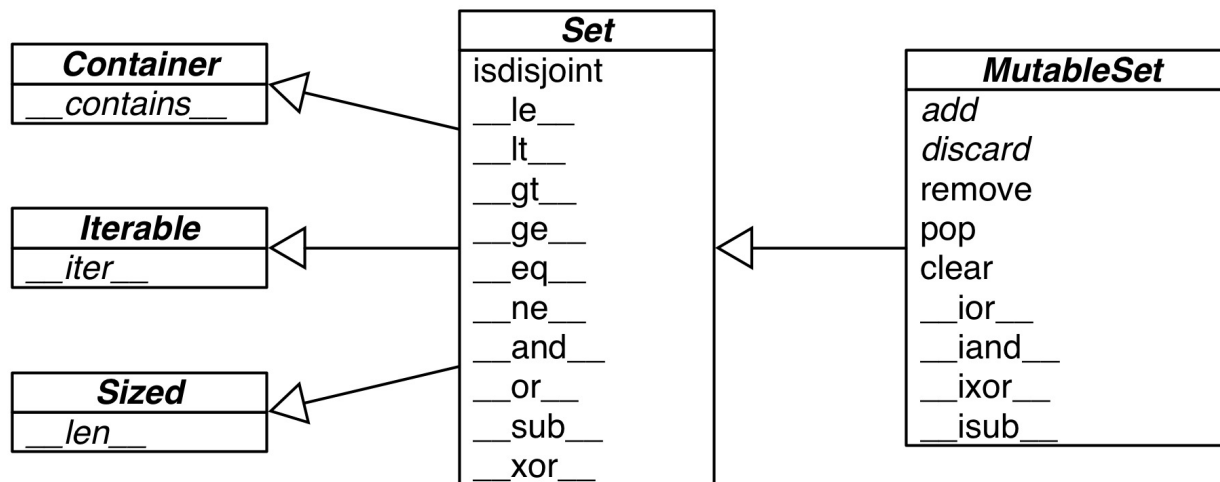
Отображения



Ограничение на ключи

- Ключ должен быть хэшируемым объектом
- Хэшируемый объект – поддерживает:
 - Метод `__hash__`
 - Метод `__eq__`
- Если объекты равны, то и их хэш-значения тоже должны быть равны.

Множества



Операции на множествах

$S \cap Z$

$s \& z$

$z \& s$

$s \&= z$

$S \cup Z$

$s | z$

$z | s$

$s |= z$

$S \setminus Z$

$s - z$

$z - s$

$s -= z$

$S \Delta Z$

$s \wedge z$

$z \wedge s$

$s \wedge= z$

$e \in S$

$e \text{ in } s$

$S \subseteq Z$

$s \leq z$

$S \subset Z$

$s < z$

$S \supseteq Z$

$s \geq z$

$S \supset Z$

$s > z$