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

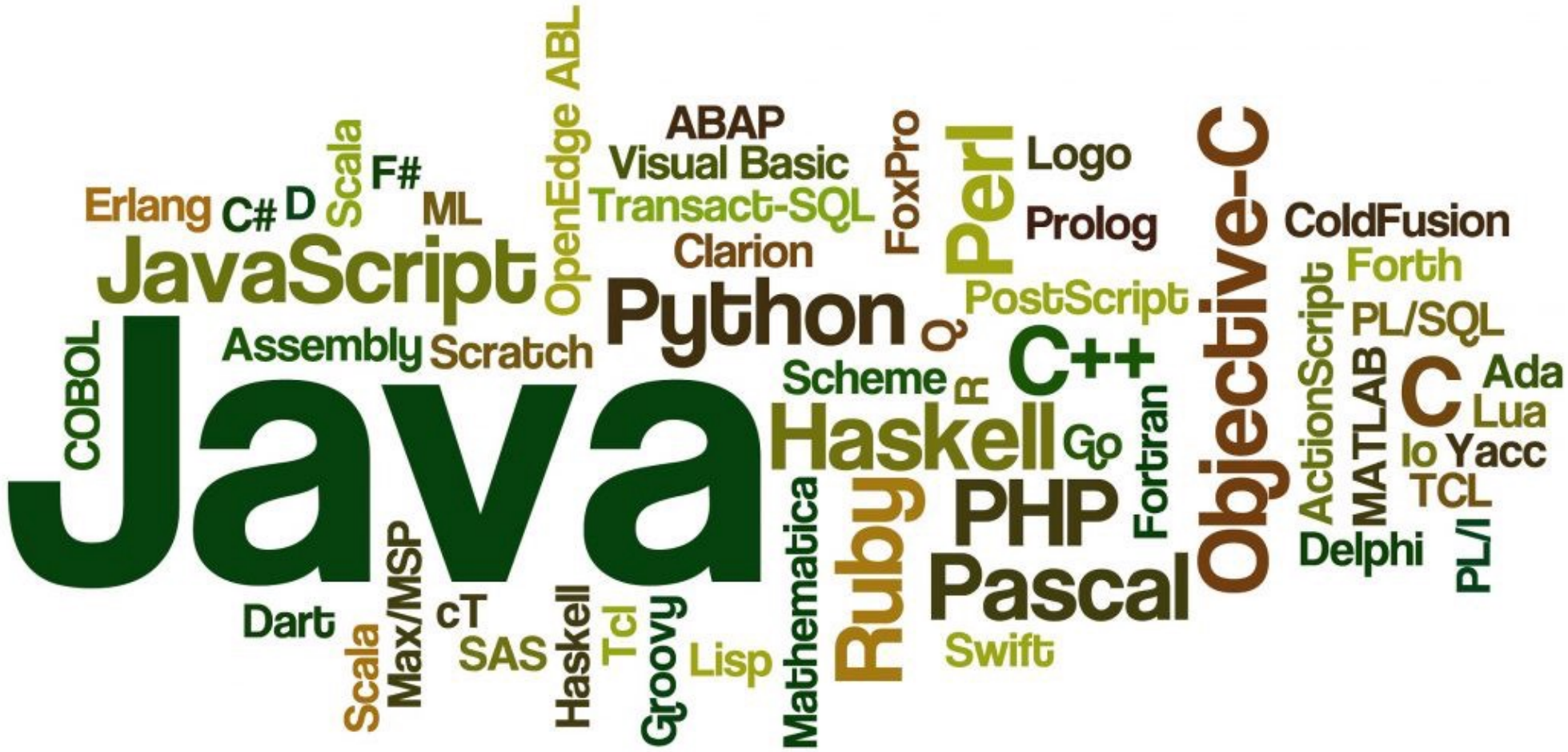
Pandas

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

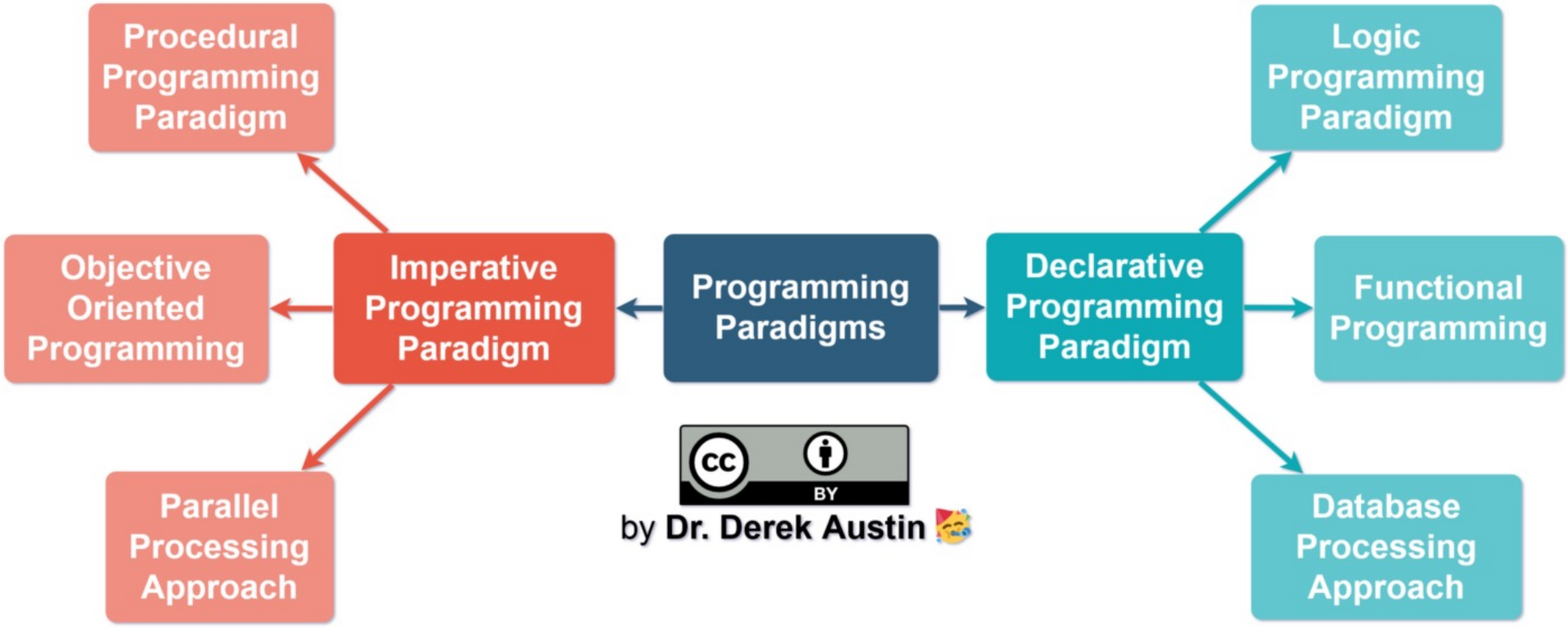
- Git

- Парадигмы

Оно нам надо?

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

Парадигмы


Procedural Programming Paradigm
Logic Programming Paradigm
Objective Oriented Programming
Imperative Programming Paradigm
Programming Paradigms
Declarative Programming Paradigm
Functional Programming
Parallel Processing Approach
Database Processing Approach
BY
by Dr. Derek Austin

Давайте решим задачу

- Дан список списков: [[1,2,3,…],[4,5,7,…],…]
- Найти сумму вторых элементов всех вложенных списков: 2+5+…

- 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
- Мультипарадигменый
- 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-ориенти́рованное программи́рование (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 k лучше?
  - Показ решения
- Обсуждение
  - Какое решение лучше и почему?

Модель данных Python

Jupyter

Переменные

- Переменные - это не ящики, это этикетки

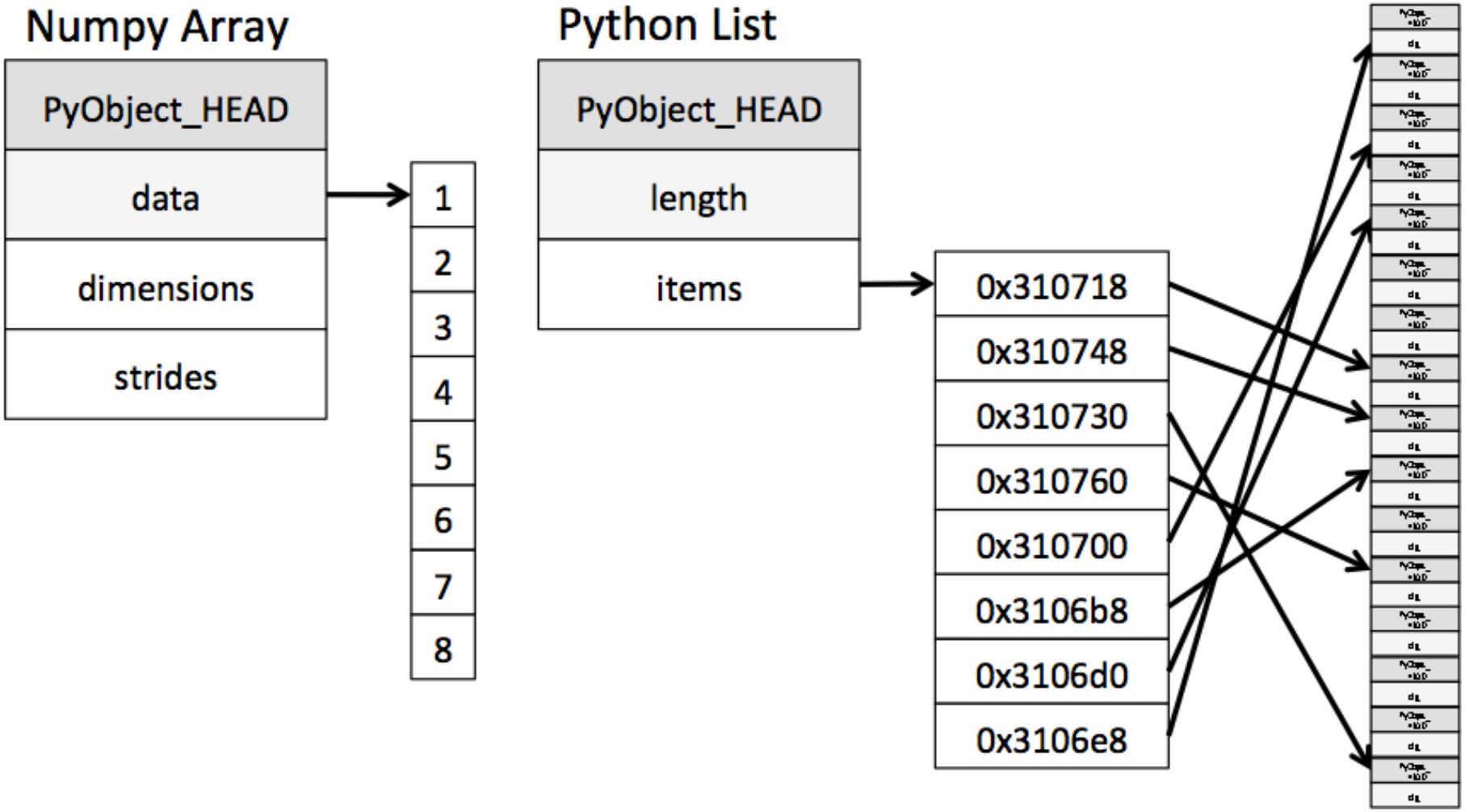
Jupyter….

Виды последовательностей

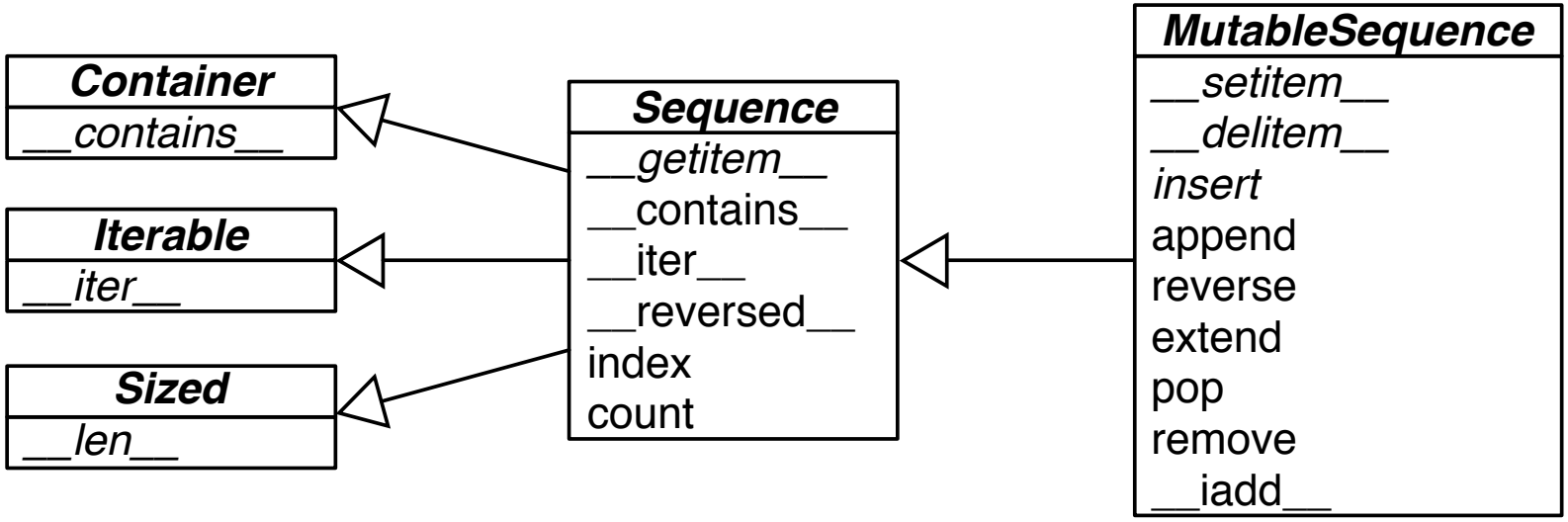
- Контейнерные:
  - list, tuple, collections.deque
- Плоские:
  - str, bytes, bytearray, memoryview, array.array

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

Представление в памяти



Последовательности



Центральность в графе

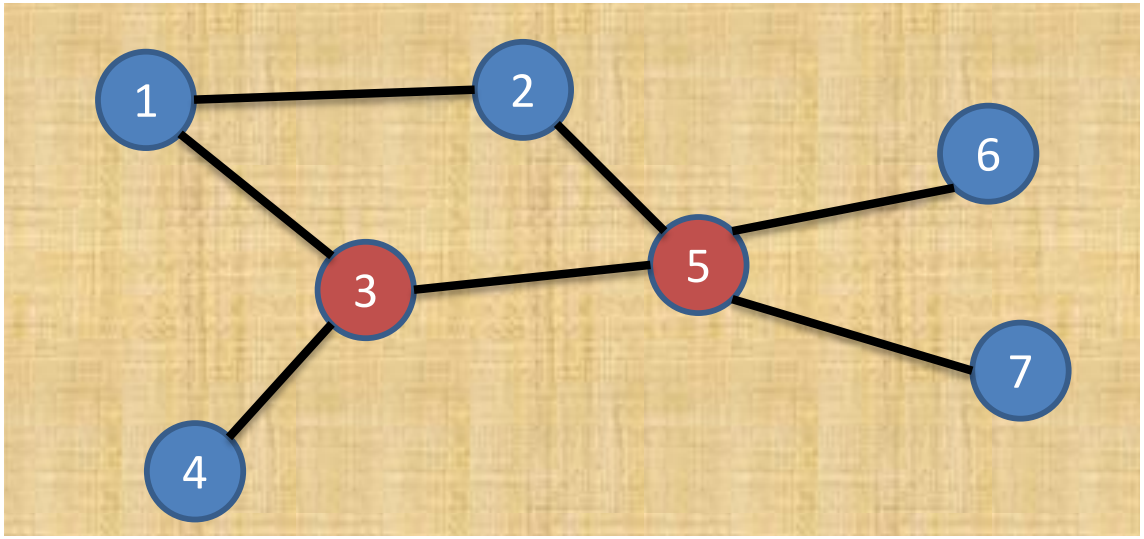
Центральность вершин в графе – это вектор, сопоставляющей каждой вершине графа некоторое число (индекс).

Наиболее распространенные индексы:

- Степенная центральность (degree centrality);
- Центральность по близости (closeness centrality);
- Центральность по посредничеству (betweenneess centrality);
- Центральность по собственному вектору (eigenvector centrality);
- Центральность PageRank.

Центральность по близости

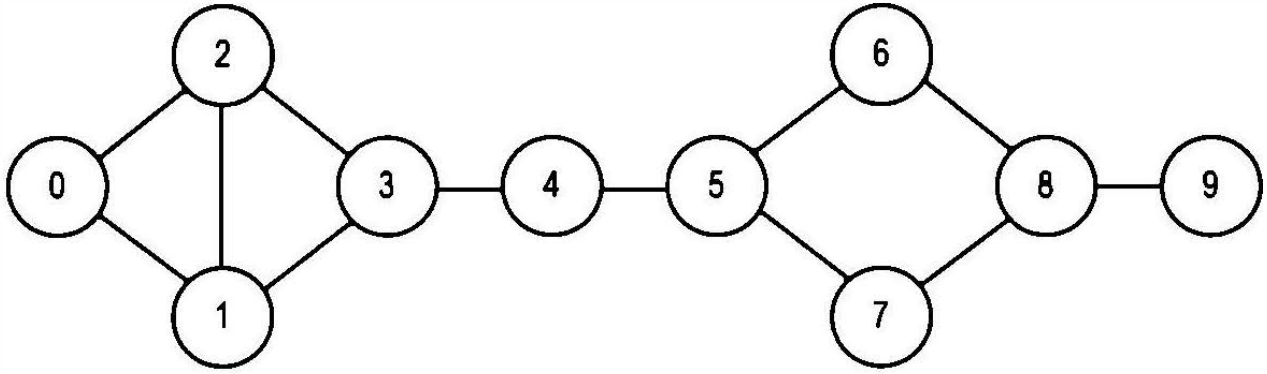
Вершина, находящаяся ближе всех к другим вершинам сети, является наиболее центральной



$$C_i = \frac{1}{\sum_j d_{ij}} \; C_i = \sum_j \frac{1}{d_{ij}}$$

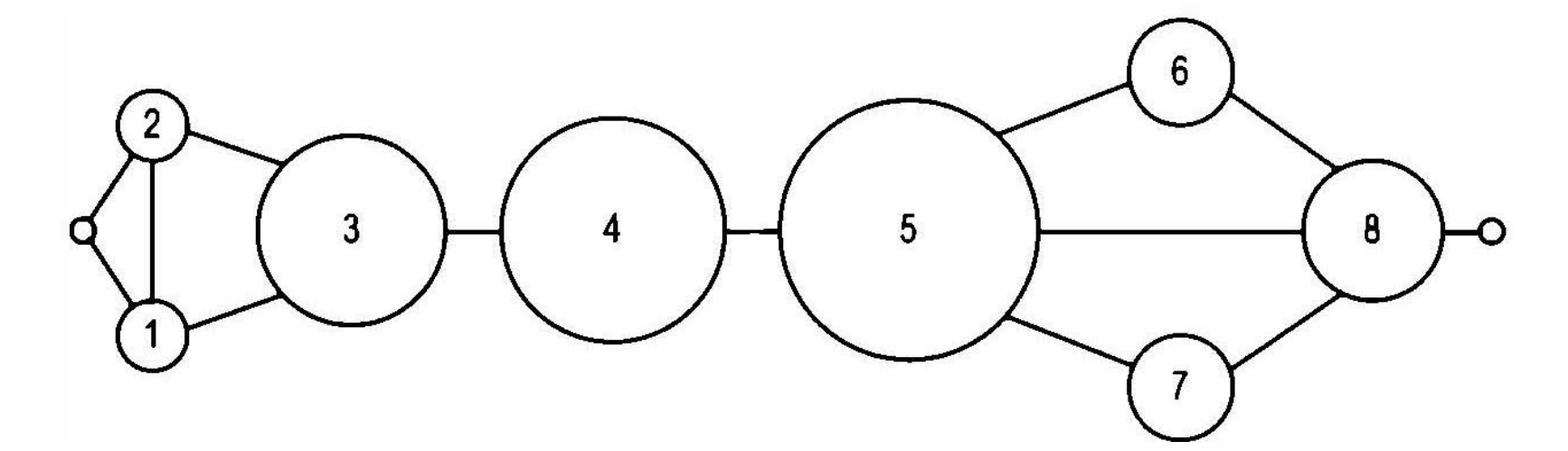
Центральность по посредничеству

Вершина, через которую проходит наибольшее число кратчайших путей, является наиболее центральной.



$$C_i = \sum_{jk} \frac{w_{jk}(i)}{w_{jk}}$$

Центральность по посредничеству



Центральность по собственному значению

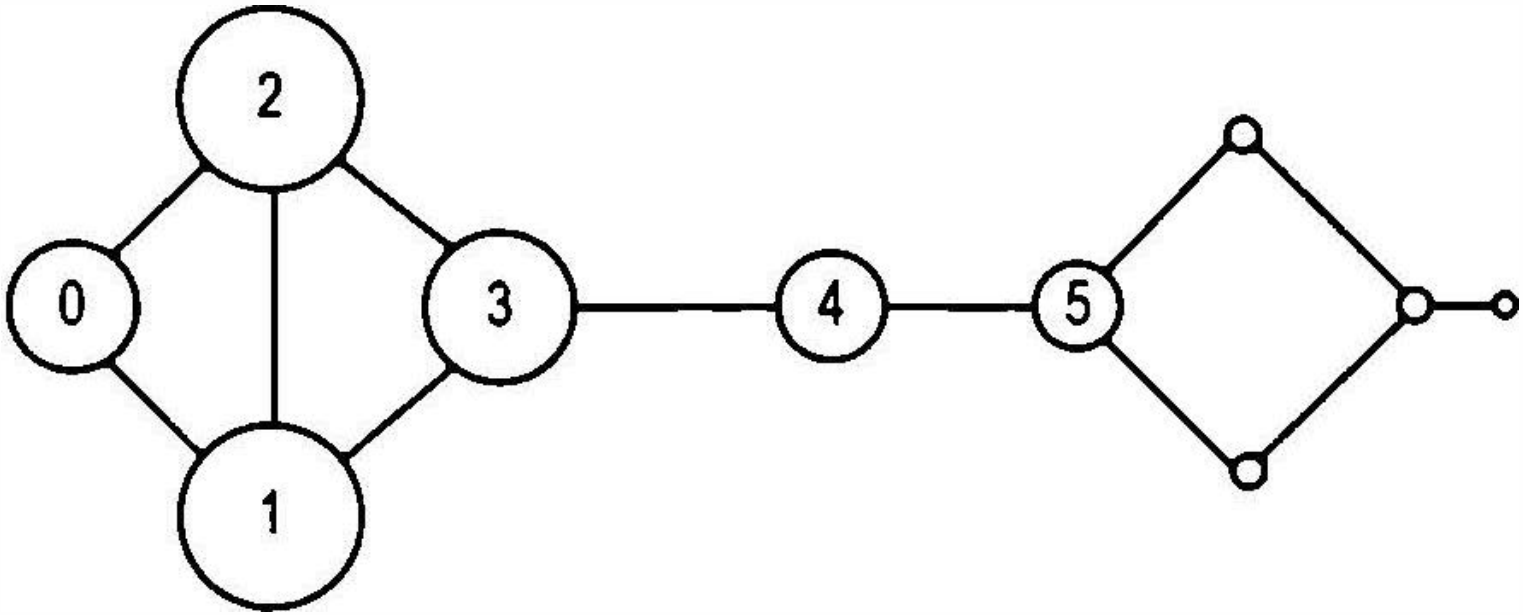
Центральность вершины i зависит от центральностей соседей вершины i.

$$x_i = \frac{1}{\lambda} \sum_{j \in F_i} x_j = \frac{1}{\lambda} \sum_{j} a_{ij} x_j$$

$$\lambda x = Ax$$

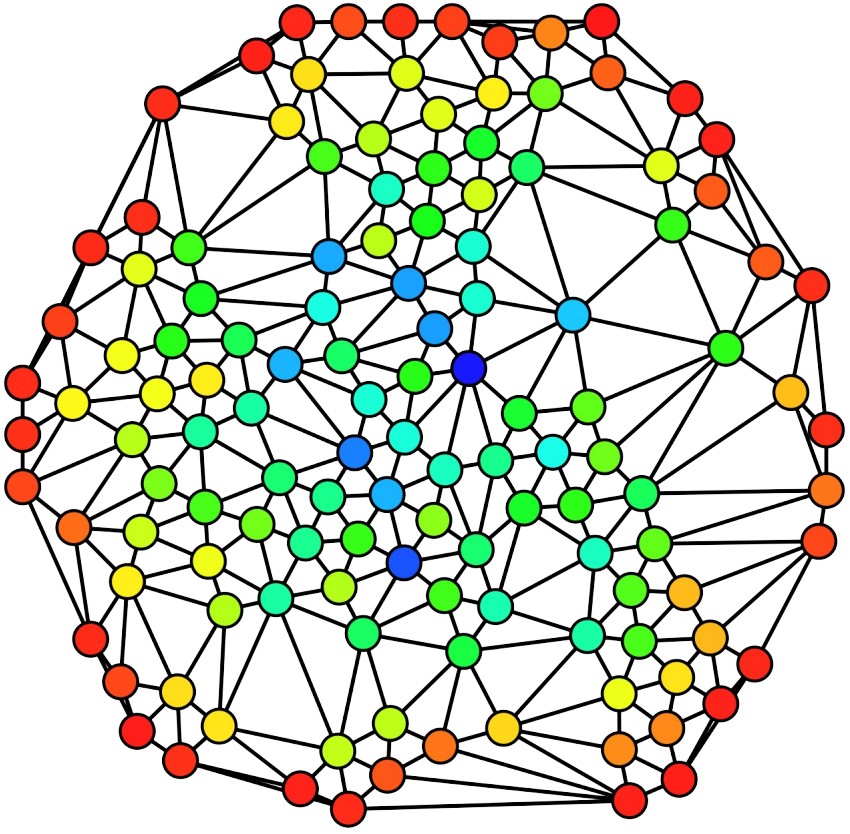
- Выбирается собственный вектор, соответствующий максимальному собственному значению.
- Данная центральность учитывает дальние взаимодействия.
- Наиболее центральными считаются вершины, которые сами указывают на сильные вершины.

Центральность по собственному значению



Задача

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



Контрольная работа

- Реализовать конечный автомат

Начальное состояние автомата (вектор длины n):

0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

- **Вход**: *целое число* – определяет поведение автомата

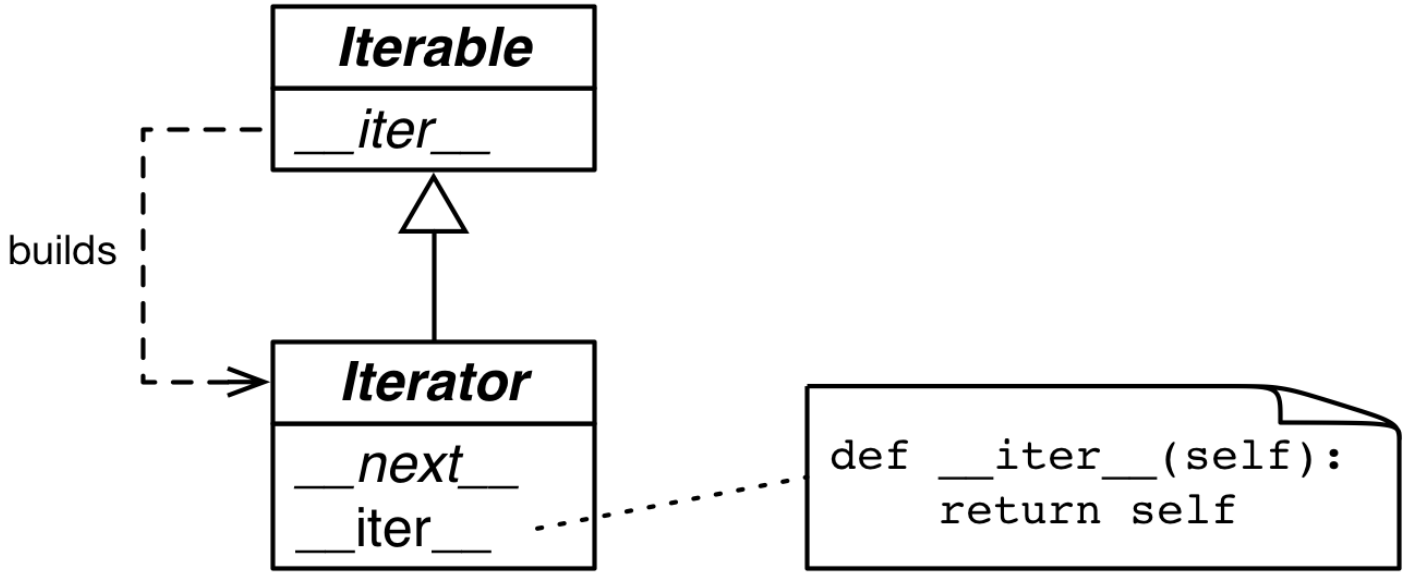
При преобразовании в бинарный вид, например числа 1:

111	110	101	100	011	010	001	000
0	0	0	0	0	0	0	1

Нижняя строка показывает какое число нужно записать в среднюю ячейку при данном состоянии трех соседних элементов вектора.

- **Выход**: последовательность состояний вектора (одна строка – одно состояние). Выводится не более m состояний

Итератор


Iterable
__iter__
builds
Iterator
__next__
__iter__
def __iter__(self):
return self

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 *

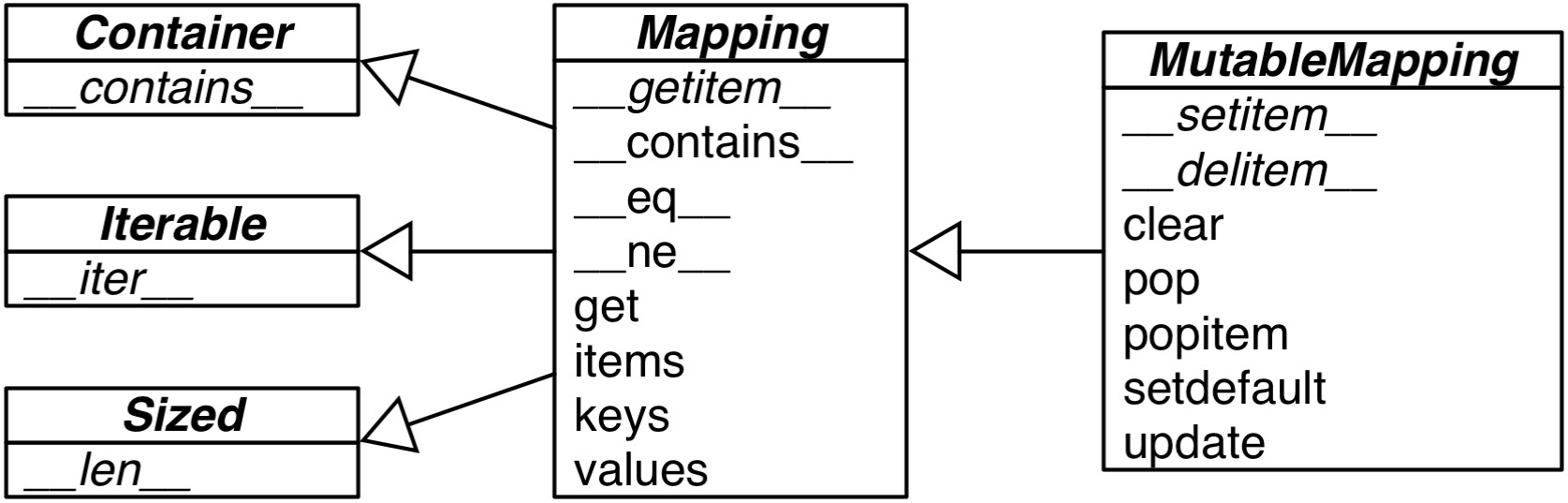
Домашнее задание

- Реализовать модель Шеллинга (модель расовой сегрегации)
- Дан квадрат n x 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.

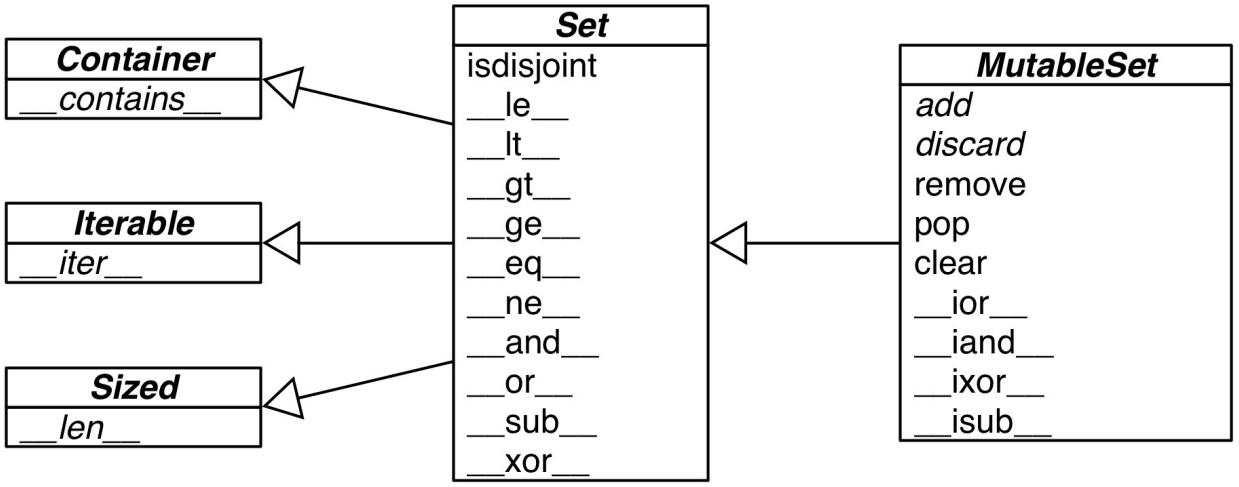
Отображения



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


Container
__contains__
Iterable
__iter__
Sized
__len__
Set
isdisjoint
__le__
__lt__
__gt__
__ge__
__eq__
__ne__
__and__
__or__
__sub__
__xor__
MutableSet
add
discard
remove
pop
clear
__ior__
__iand__
__ixor__
__isub__

Операции на множествах

$S \cap Z$	`s & z`	$S \setminus Z$	`s - z`		
	`z & s`		`z - s`	$e \in S$	`e in s`
	`s &= z`		`s -= z`	$S \subseteq Z$	`s <= z`
				$S \subset Z$	`s < z`
$S \cup Z$	`s | z`	$S \Delta Z$	`s ^ z`	$S \supseteq Z$	`s >= z`
	`z | s`		`z ^ s`		
	`s |= z`		`s ^= z`	$S \supset Z$	`s > z`