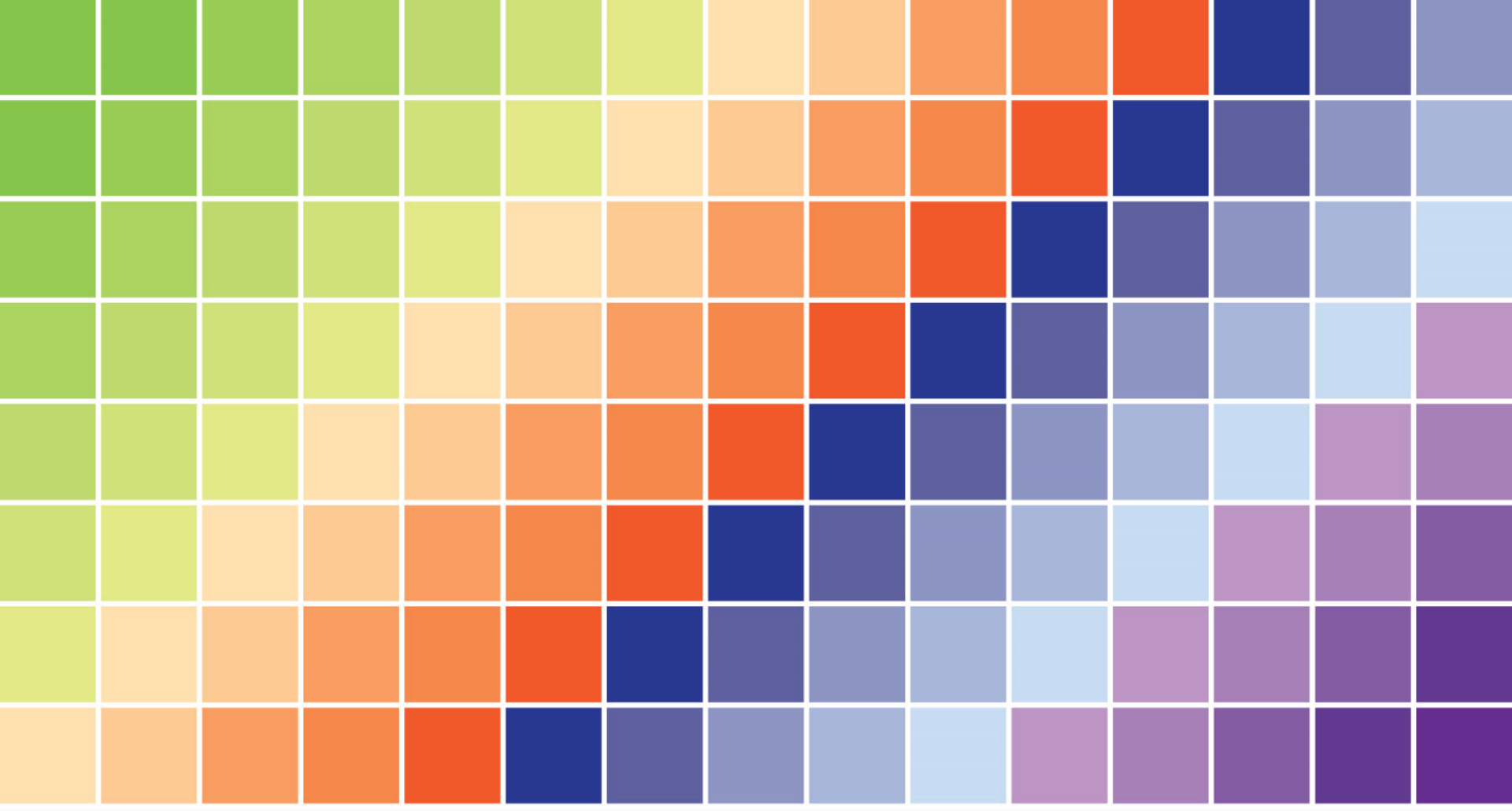


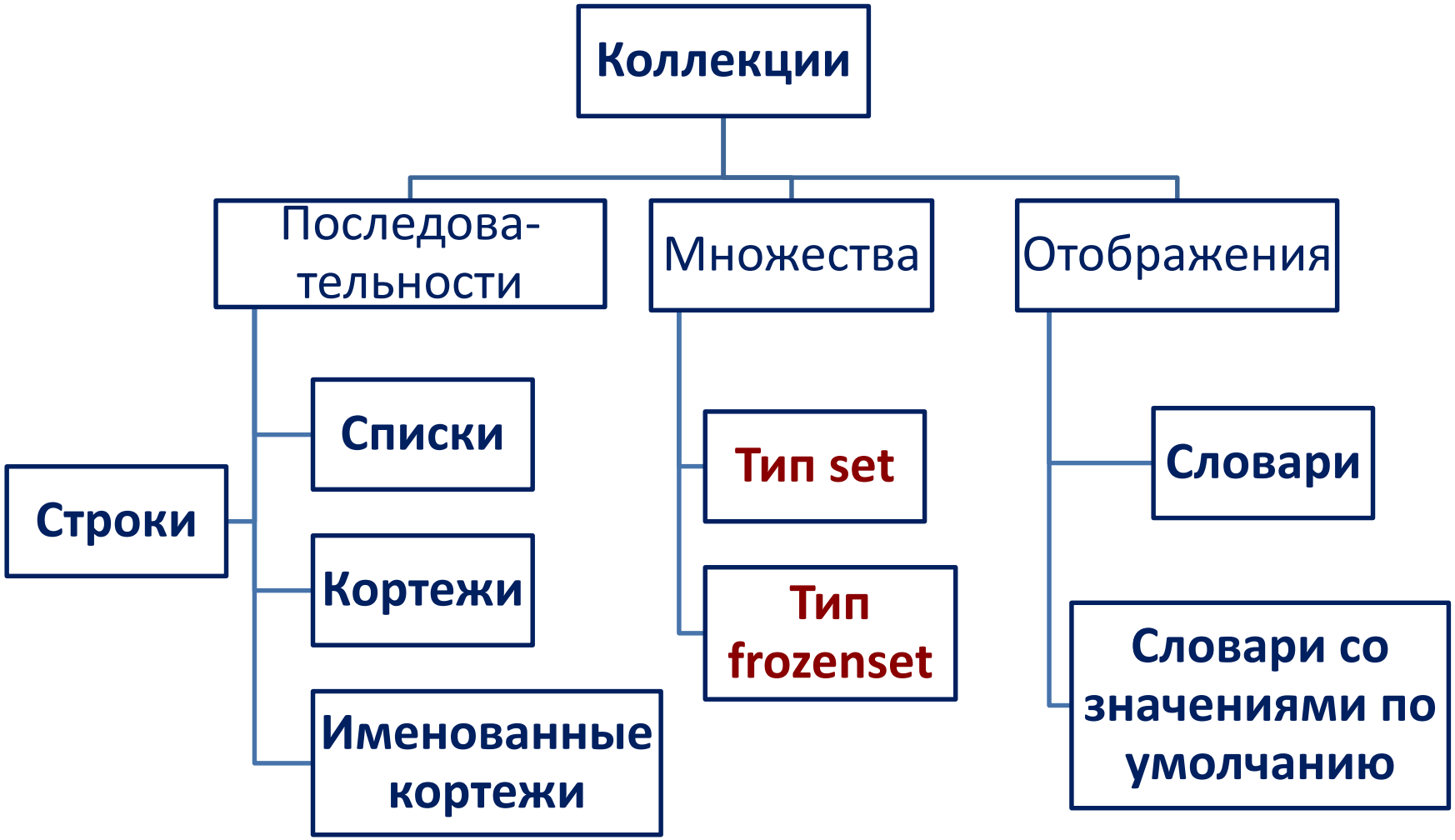
School of Computer Science

МНОЖЕСТВА. ИТОГИ ПО КОЛЛЕКЦИЯМ

ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы

ТИПЫ КОЛЛЕКЦИЙ В PYTHON



МНОЖЕСТВО (SET)

- Составной тип данных, поддерживающий следующие операции:
  - Проверки на вхождение in и not in
  - Определение размера len(object)
  - **Содержит только уникальные элементы**
  - Порядок элементов не важен и не имеет значения
- Множество, как список и словарь, это **изменяемый** тип данных
- Но множество может содержать только элементы **неизменяемых** типов

МНОЖЕСТВА

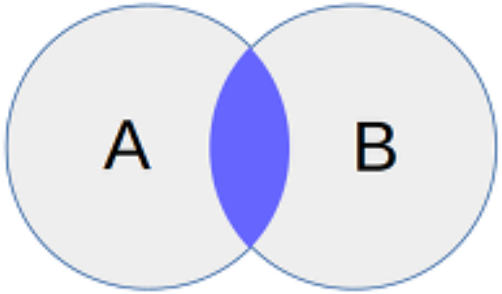
- Объявление множества :
  - set()
  - { значения через запятую }

- Примеры:
  - `set1 = set()`
  - `set2 = {2}`
  - `set3 = {"a", "b", "c", "d"}`
  - `set4 = {7, "sun", ('x', 11), -0.15}`
  - `set5 = set("hello")`

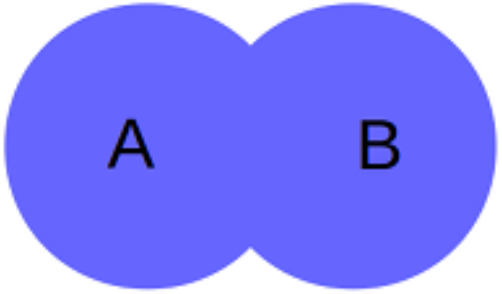
ОПЕРАЦИИ С МНОЖЕСТВАМИ

Пересечение, объединение и разница множеств

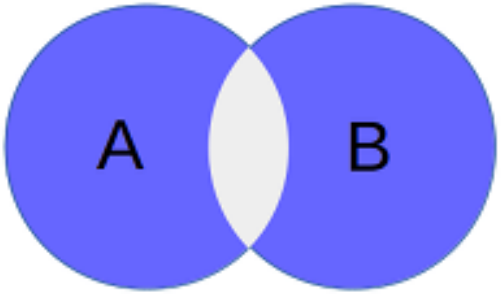
Пересечение A&B



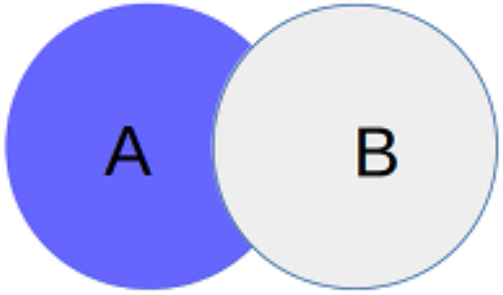
Объединение A|B



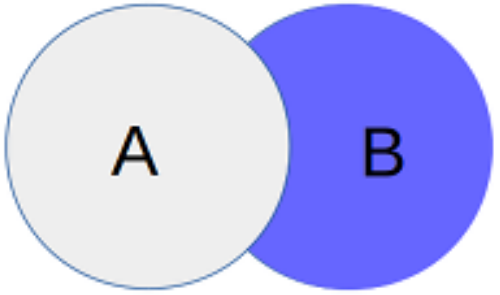
Симметричная разница A^B



Разница A-B



Разница B-A



МНОЖЕСТВА.
ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ

- Регистрация уникальных объектов
- Быстрая проверка на принадлежность
- Устранение дубликатов из последовательности
- Выполнение математических операций теории множеств

МЕТОДЫ МНОЖЕСТВ

Вызов	Описание
s.add(x)	Добавляет элемент x во множество s, если x отсутствует в s
s.clear()	Удаляет все элементы из множества s
s.copy()	Возвращает копию множества s*
s.difference(t), s - t	Возвращает новое множество, включающее элементы множества s, которые отсутствуют в множестве t*
s.difference_ update(t), s -= t	Удаляет из множества s все элементы, присутствующие в множестве t
s.discard(x)	Удаляет элемент x из множества s; смотрите также метод set.remove()

МЕТОДЫ МНОЖЕСТВ. ПЕРЕСЕЧЕНИЕ

Вызов	Описание
s.intersection(t), s & t	Возвращает множество с элементами, присутствующими одновременно в множествах s и t*
s.intersection_ update(t), s &= t	Оставляет во множестве s пересечение множеств s и t
s.isdisjoint(t), s != t	Возвращает True, если множества s и t не имеют общих элементов (т.е. не пересекаются)*
s.issubset(t), s <= t	Возвращает True, если s==t или s подмножество t; используйте s < t, чтобы проверить, является ли множество s только подмножеством t*

МЕТОДЫ МНОЖЕСТВ

Вызов	Описание
s.issuperset(t), s >= t	Возвращает True, если s==t или является его надмножеством; чтобы проверить, что s только надмножество t, следует использовать проверку s > t*
s.pop()	Возвращает и удаляет случайный элемент множества s или возбуждает исключение KeyError, если s – это пустое множество
s.remove(x)	Удаляет элемент x из множества s или возбуждает исключение KeyError, если элемент x отсутствует в множестве s; смотрите также метод set.discard()

МЕТОДЫ МНОЖЕСТВ «ИСКЛЮЧАЮЩЕЕ ИЛИ»

Вызов	Описание
s.symmetric_difference(t), s ^ t	Возвращает новое множество, включающее все элементы, из s и t, за исключением тех элементов, которые присутствуют в обоих множествах одновременно*
s.symmetric_difference_update(t), s ^= t	Возвращает в множестве s результат строгой дизъюнкции множеств s и t

МЕТОДЫ МНОЖЕСТВ. ОБЪЕДИНЕНИЕ

Вызов	Описание
s.union(t), s | t	Возвращает новое множество, включающее все элементы множества s и все элементы множества t, отсутствующие в множестве s*
s.update(t), s |= t	Добавляет во множество s все элементы множества t, отсутствующие в множестве s. *Метод update() и другие заменители операций позволяют в параметре t передавать, в том числе, последовательности других типов*

МЕТОДЫ МНОЖЕСТВ. ПРИМЕРЫ

- См. создание множеств в set_creation.py
- См. объединение множеств в set_union.py
- См. пересечение множеств в set_intersection.py
- См. разность множеств в set_difference.py
- См. симметрическую разницу множеств в set_xor.py

ФИКСИРОВАННОЕ МНОЖЕСТВО

- Инициируются с помощью вызова frozenset()
- Не поддерживают внесение изменений после инициации
- Поддерживают некоторые операции обычных множеств (* в таблицах выше)
- Для списка есть кортеж, а для множества frozenset, чтобы использовать там, где нужны неизменяемые объекты

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ.
ПОИСК ПЕРЕСЕЧЕНИЙ В 2-УХ СПИСКАХ

Имеется 2 списка:

```
>>> list_a = [5, 2, 3,'r',4,'ee', 8, 11, 23]
>>> list_b = [4, 1,'we','ee', 2,'r', 3, -6, 0.23]
```

Алгоритм определения пересекающихся значений:

```
>>> match_list = []
>>> for i in list_a:
        for j in list_b:
            if i == j:
                match_list.append(i)
                break
```

Как его переписать в одну строку?

```
>>> match_list
[2, 3, 'r', 4, 'ee']
```

ПРИМЕР
ССЫЛКИ НА СТОРОННИЕ САЙТЫ В HTML

- Задача: поиск ссылок на сторонние сайты в HTML-файлах

- Решение:
  - Итерации по файлам
  - Итерации по строкам в каждом файле
  - Поиск метки http:// столько раз, сколько она встречается в каждой строке
  - Использование словаря и множества для сохранения результатов

- См. скрипт external_sites.py

СОЗДАНИЕ МНОЖЕСТВА

Какой способ создать множество является недопустимым:

1. a_set = {}
2. a_set = {1}
3. a_set = {1, 2}
4. a_set = {(1,),(2,)}
5. a_set = set(list(tuple([1]*10)))

???

ВОПРОСЫ
УДАЛЕНИЕ ИЗ МНОЖЕСТВА

a_set = {1, 2, 3, 4, 5}

Какой вызов не породит исключение:

1. a_set.remove(6)
2. a_set.delete(6)
3. a_set.erase(6)
4. a_set.discard(6)

???

ВОПРОСЫ
ДОБАВЛЕНИЕ В МНОЖЕСТВО

Как добавить значения из списка upd_list в множество a_set:

1. a_set.insert(upd_list)
2. a_set.add(upd_list)
3. a_set.update(upd_list)
4. a_set.append(upd_list)

???

ВОПРОСЫ
ИТЕРАЦИЯ ПО МНОЖЕСТВУ

Сколько букв напечатает этот код?

```
>>> for letter in set('apple'):
        print(letter)
```

В каком порядке?

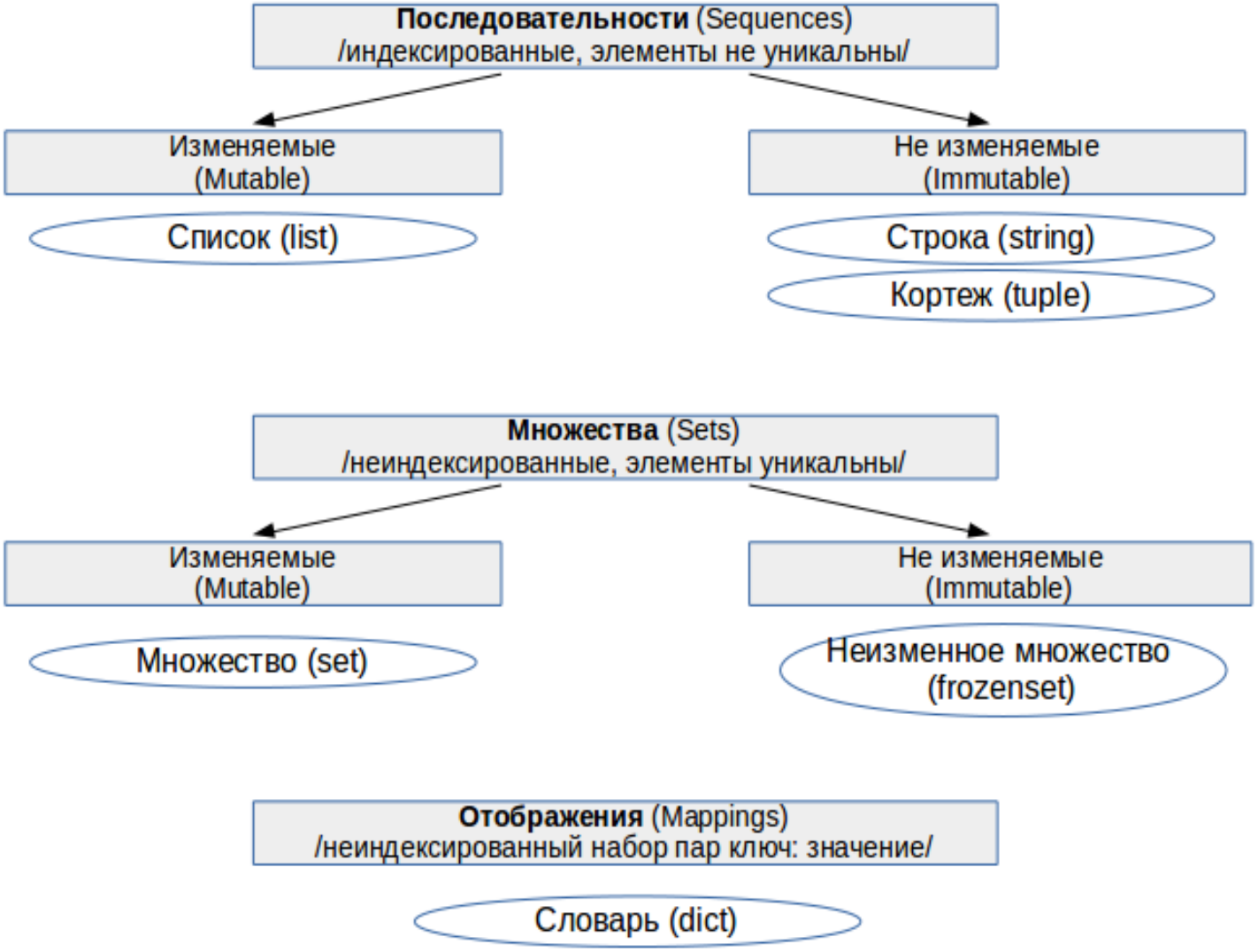
ДЛИНА МНОЖЕСТВА

Какова длина множества number_set?

```
>>> number_set = {1, 2, 3, 3, 4, 5}
>>> len(number_set)
```

???

КЛАССИФИКАЦИЯ КОЛЕЛКЦИЙ



ОБОБЩЕНИЕ СВОЙСТВ КОЛЛЕКЦИЙ.

Тип	Изменя-емость	Индекси-рованность	Уникаль-ность	Как создаем
Список (list)	+	+	-	[] list()
Кортеж (tuple)	-	+	-	(, , ,) tuple()
Строка (str)	-	+	-	'' ""
Множество (set)	+	-	+	{item1, item2} set()
Фикс. множество (frozenset)	-	-	+	frozenset()
Словарь (dict)	+ элементы, - ключи, + значения	-	+ элементы, + ключи, - значения	{} {key:value, } dict()

КОЛЛЕКЦИИ.
ОБЩИЕ ФУНКЦИИ

Вызов	Описание
s + t	Конкатенация последовательностей s и t
s * n	Конкатенация из int n последовательностей s
x in i	True, если элемент x присутствует в итерируемом объекте i, обратная проверка выполняется с помощью оператора not in
all(i)	True, если все элементы итерируемого объекта i в логическом контексте оцениваются как значение True
any(i)	True, если хотя бы один элемент итерируемого объекта i в логическом контексте оценивается как значение True

КОЛЛЕКЦИИ.
ОБЩИЕ ФУНКЦИИ ДЛЯ ИТЕРАТОРОВ

Вызов	Описание
enumerate (i [, start])	Получение последовательности кортежей (index, item), где значения индексов отсчитываются от 0 или от значения start
len(x)	Количество элементов в коллекции или количество символов в строке
max(i [, key]) **min(i [, key])**	Возвращает наибольший/наименьший элемент в итерируемом объекте i или элемент с наибольшим/наименьшим значением key(item), если функция key определена

КОЛЛЕКЦИИ.
ОБЩИЕ ФУНКЦИИ ДЛЯ ИТЕРАТОРОВ

Вызов	Описание
reversed(s)	Возвращает элементы последовательности s в обратном порядке
sorted(i [, key, reverse])	Возвращает список элементов итератора i в отсортированном порядке; аргумент key используется для задания порядка сортировки. Если аргумент reverse имеет значение True, сортировка выполняется в обратном порядке
sum(i [, start])	Возвращает сумму элементов итерируемого объекта i, плюс аргумент start (по умолчанию = 0); объект i не должен содержать строк

ПРИМЕРЫ.
ФУНКЦИИ ИТЕРАБЕЛЬНЫХ ОБЪЕКТОВ

- См. пример использования групповых функций в all_any_len_min_max_sum.py
- См. примеры сортировок в sorted_reversed.py
- См. пример поиска слова в текстовых файлах в скрипте grep_word.py, пример на использование функции enumerate()

СПАСИБО ЗА ВНИМАНИЕ !

ВОПРОСЫ ?

School of Computer Science