

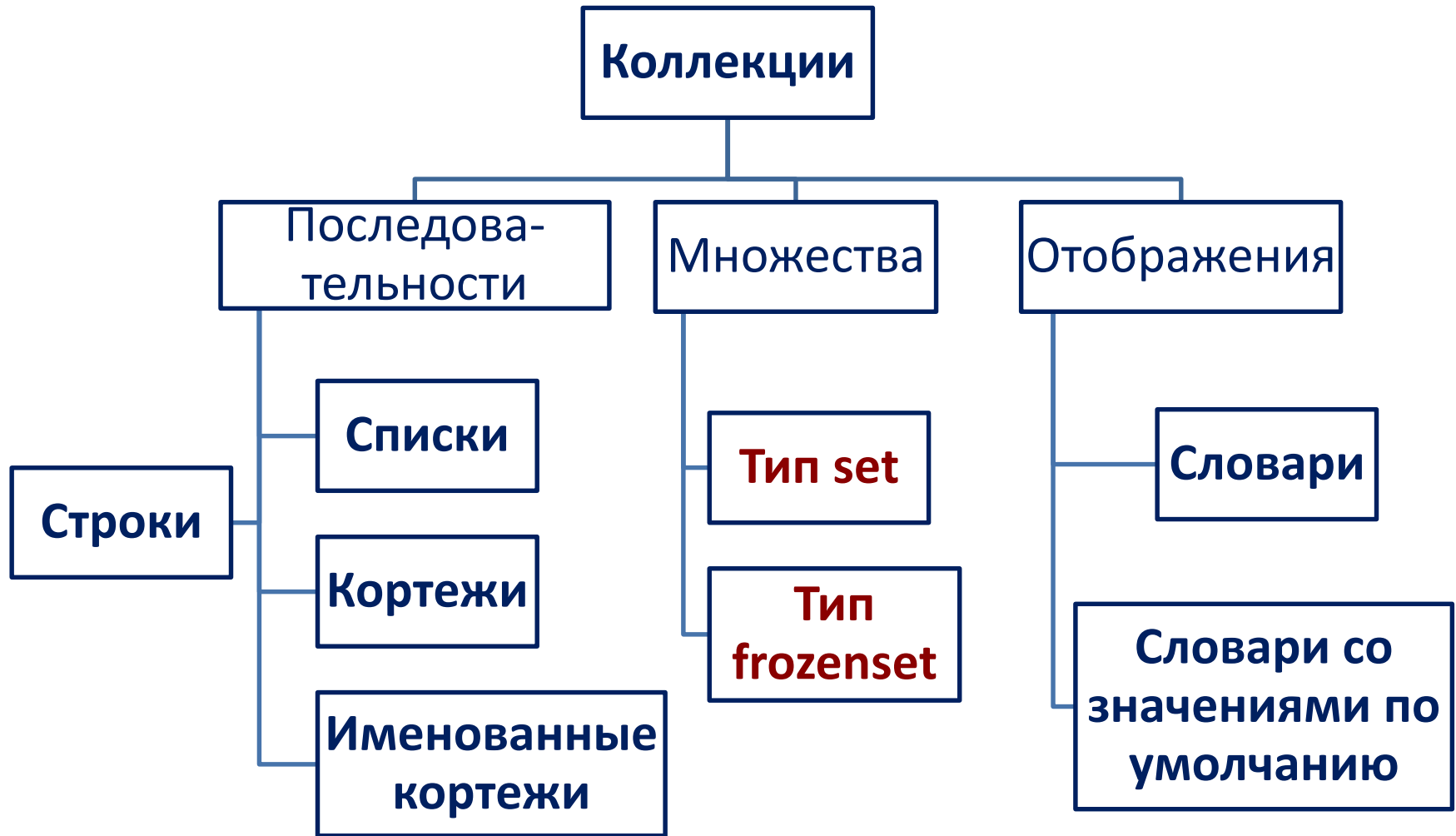
*School of
Computer
Science*

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

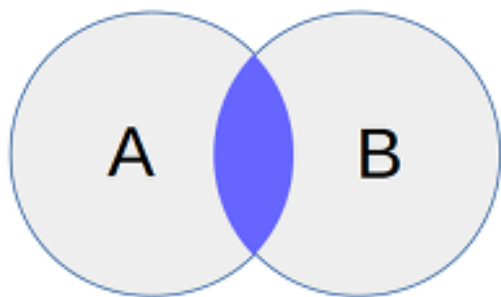
- Объявление множества :
 - `set()`
 - `{ значения через запятую }`
- Примеры:
 - `set1 = set()`
 - `set2 = {2}`
 - `set3 = {"a", "b", "c", "d"}`
 - `set4 = {7, "sun", ('x', 11), -0.15}`
 - `set5 = set("hello")`



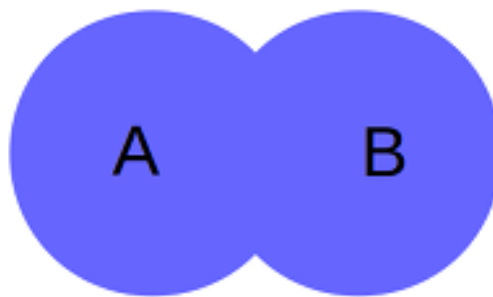
ОПЕРАЦИИ С МНОЖЕСТВАМИ

Пересечение, объединение и разница множеств

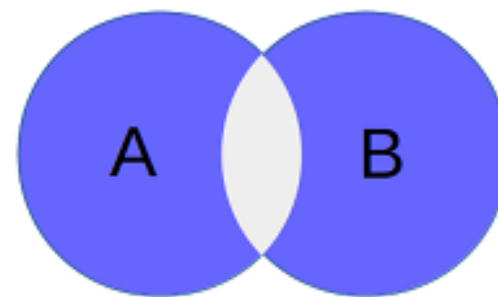
Пересечение $A \cap B$



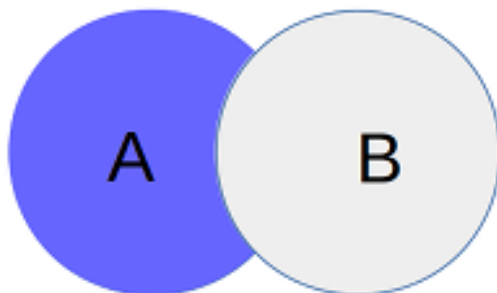
Объединение $A \cup B$



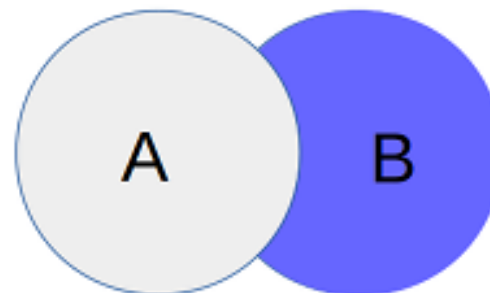
Симметричная разность $A \oplus B$



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



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



МЕТОДЫ МНОЖЕСТВ «ИСКЛЮЧАЮЩЕЕ ИЛИ»

Вызов

Описание

**s.symmetric_
difference(t),
 $s \wedge t$**

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),
 $s \wedge= t$**

Возвращает в множестве s результат строгой дизъюнкции множеств s и t



МЕТОДЫ МНОЖЕСТВ. ОБЪЕДИНЕНИЕ

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



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

- Иницииируются 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']
```



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`



ВОПРОСЫ

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?



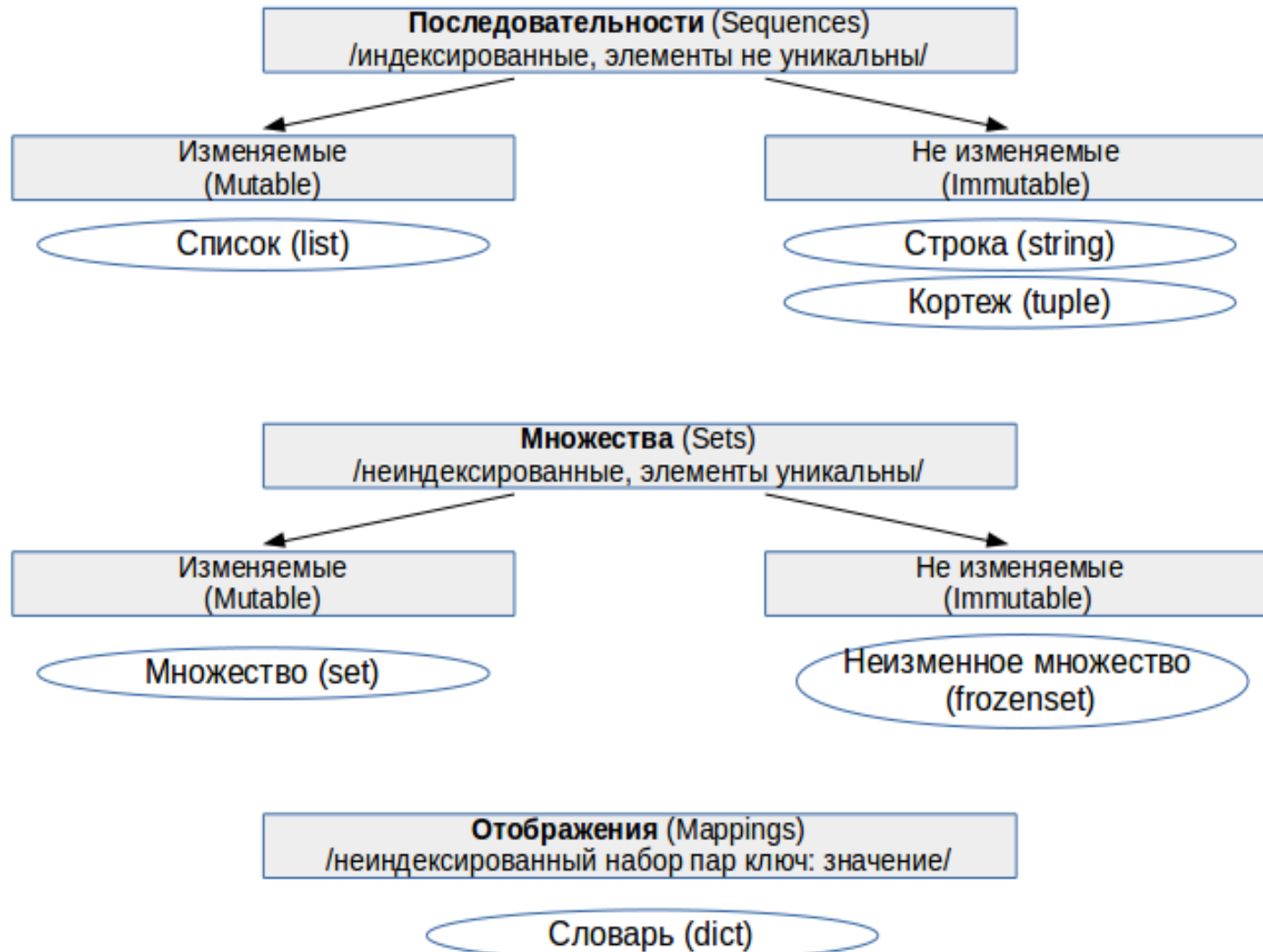
Какова длина множества `number_set`?

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

???



КЛАССИФИКАЦИЯ КОЛЛЕКЦИЙ





ОБОБЩЕНИЕ СВОЙСТВ КОЛЛЕКЦИЙ.

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



КОЛЛЕКЦИИ.

ОБЩИЕ ФУНКЦИИ ДЛЯ ИТЕРАТОРОВ

Вызов	Описание
<code>enumerate (i [, start])</code>	Получение последовательности кортежей (index, item), где значения индексов отсчитываются от 0 или от значения start
<code>len(x)</code>	Количество элементов в коллекции или количество символов в строке
<code>max(i [, key])</code> <code>min(i [, key])</code>	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
<code>reversed(s)</code>	Возвращает элементы последовательности <code>s</code> в обратном порядке
<code>sorted(i [, key, reverse])</code>	Возвращает список элементов итератора <code>i</code> в отсортированном порядке; аргумент <code>key</code> используется для задания порядка сортировки. Если аргумент <code>reverse</code> имеет значение <code>True</code> , сортировка выполняется в обратном порядке
<code>sum(i [, start])</code>	Возвращает сумму элементов итерируемого объекта <code>i</code> , плюс аргумент <code>start</code> (по умолчанию = 0); объект <code>i</code> 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()`



ПРАКТИЧЕСКАЯ РАБОТА

ПОДСЧЁТ СРЕДНЕГО РОСТА УЧАЩИХСЯ

Дан файл **children.txt** в формате TSV с информацией о росте школьников разных классов. Напишите программу, которая прочитает этот файл и подсчитает для каждой параллели средний рост учащихся.

Файл состоит из набора строк, каждая из которых представляет собой три поля, разделённых символом табуляции:

Класс	Фамилия	Рост
-------	---------	------

Класс содержит буквенные модификаторы, которые необходимо отбросить. Номер класса может быть от 1 до 11 включительно. В фамилии нет пробелов, а в качестве роста используется натуральное число.

Полученные данные необходимо записать в файл. Выводить информацию о среднем росте следует в порядке возрастания номера класса (для классов с первого по одиннадцатый), рост записать с точностью до двух знаков после запятой. Если про какой-то класс нет информации, необходимо вывести напротив него прочерк:

1 : 125.18

2 : —

3 : 131.63

...

11 : 175.50

СПАСИБО ЗА ВНИМАНИЕ !
ВОПРОСЫ ?



*School of
Computer
Science*