



*School of
Computer
Science*

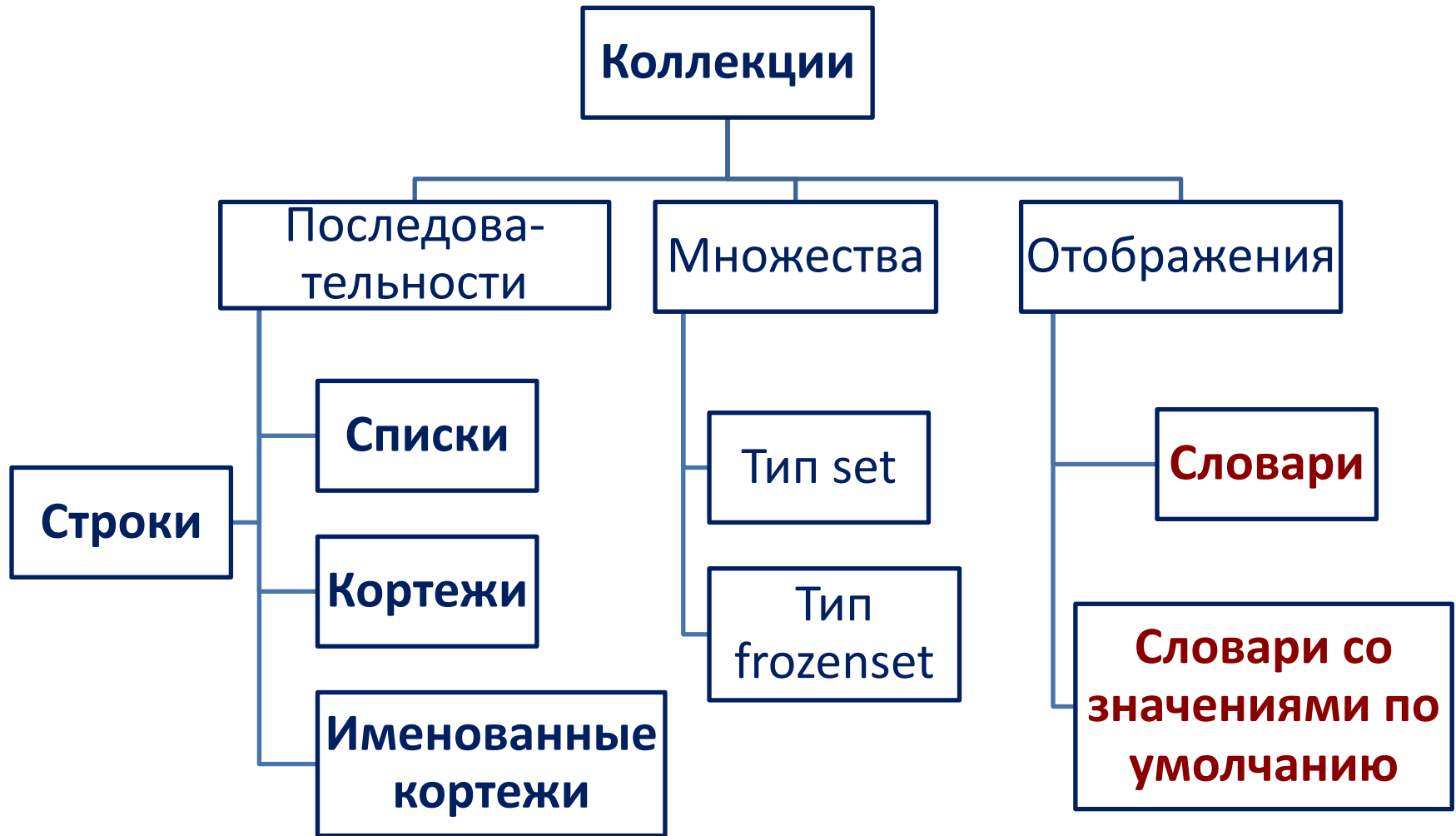
СЛОВАРИ

ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы



ТИПЫ КОЛЛЕКЦИЙ В PYTHON





СЛОВАРЬ (DICT)

- Составной тип данных, поддерживающий следующие операции:
 - Проверки на вхождение `in` и `not in`
 - Определение размера `len(object)`
 - Обращение к элементу по **ключу**: `object[key]`
 - Ключ – любой **неизменяемый** тип данных
 - Итерации, не гарантирующие по умолчанию какую-либо последовательность элементов
- Словарь, как и список, это **изменяемый** тип данных

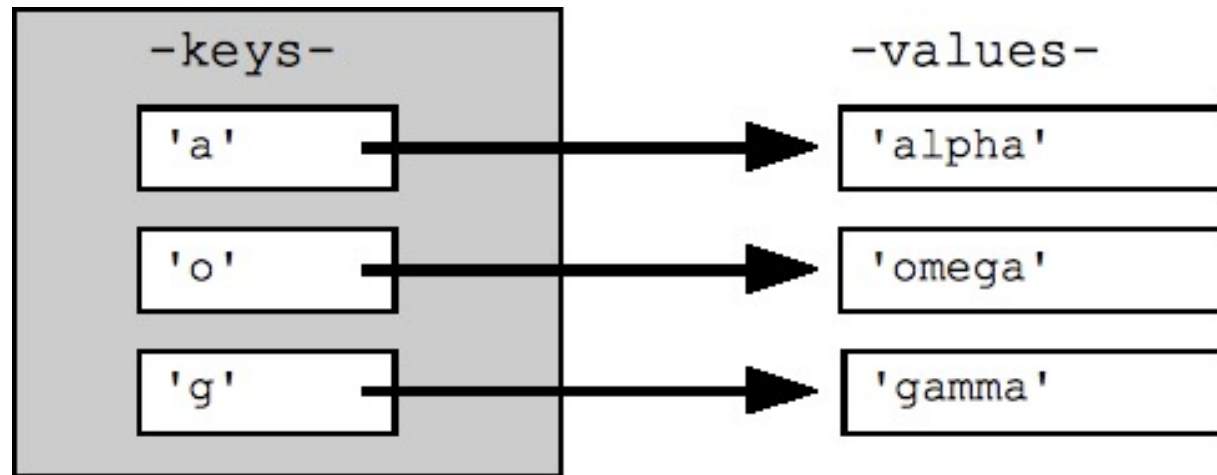


СЛОВАРИ

- Объявление словаря:
 - `dict()`
 - `{}`
 - { пары *key:value* через запятую }
- Примеры:
 - `dic1 = dict()`
 - `dic2 = {}`
 - `dic3 = {"dog": "May", "cat": "Max"}`
 - `dic4 = {"dogs": ['May', 'Chappy'], "cat": "Goblin"}`
 - `dic5 = dict(name='Serge', age=46)`



ПРЕДСТАВЛЕНИЕ СЛОВАРЯ



dict Создание словаря в Python

- Словарь – это неупорядоченная коллекция элементов (ключ, значение) с **уникальными ключами неизменяемых типов данных**
- **Значения** в словаре, в отличие от ключей, **могут иметь любой тип данных**



СЛОВАРИ.

ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ

- Быстрые манипуляции с данными по ключу
- Массив с непредсказуемыми индексами (разреженный массив)
- Использование нечисловых индексов
- Установка соответствия между объектами
- Хранение данных, связанных с каким-либо объектом по его уникальному ид.
 - Например, подсчет количества строк, слов, ...



СОПРОВОЖДЕНИЕ СЛОВАРЕЙ

- Доступ к элементам:
 - По ключу в `[]` справа от `=`
 - Ключ должен существовать
 - **`in`** – проверка существования ключа
 - С умолчанием при помощи метода **`get()`**
- Добавление/обновление элемента:
 - По ключу в `[]` слева от `=`
 - Ключ создается при первом присваивании
- Удаление элемента:
 - **`del dict[key]`**
 - **`dict.pop(key)`**



ИЗБРАННЫЕ МЕТОДЫ СЛОВАРЕЙ

Вызов	Описание
<code>d.clear()</code>	Удаляет все элементы из словаря d
<code>d.copy()</code>	Возвращает копию словаря d
<code>d.get(k [,v])</code>	Возвращает значение по ключу k. Если ключ не определен в словаре, то возвращается умолчание v. Если умолчание не задано, то при отсутствии ключа k в словаре возвращается None
<code>d.items()</code>	Возвращает набор всех пар в словаре. Каждая пара – это кортеж из 2-ух элементов: первый – ключ, второй – соответствующее ему значение



ИЗБРАННЫЕ МЕТОДЫ СЛОВАРЕЙ

Вызов	Описание
<code>d.keys()</code>	Возвращает набор всех ключей словаря
<code>d.pop(k)</code> , подобна <code>del d[k]</code>	<code>pop()</code> возвращает значение ключа <code>k</code> и удаляет его из словаря или возбуждает исключение <code>KeyError</code> , если ключ <code>k</code> не найден. <code>del d[k]</code> ничего не возвращает
<code>d.pop(k,v)</code>	Возвращает значение ключа <code>k</code> и удаляет из словаря элемент с ключом <code>k</code> или возвращает значение <code>v</code> , если ключ <code>k</code> отсутствует в словаре



ИЗБРАННЫЕ МЕТОДЫ СЛОВАРЕЙ

Вызов	Описание
d.popitem()	Возвращает и удаляет произвольную пару (ключ, значение) из словаря d или возбуждает исключение KeyError, если словарь d пуст
d.setdefault(k [, v])	То же, что и dict.get() за исключением того, что, если ключ k в словаре отсутствует, в словарь вставляется новый элемент с ключом k и со значением None или v, если аргумент v задан
d.values()	Возвращает набор всех значений из словаря d



ИЗБРАННЫЕ МЕТОДЫ СЛОВАРЕЙ

Вызов	Описание
d.update(a)	Добавляет в словарь d пары (ключ, значение) из a, которые отсутствуют в словаре d, а для каждого ключа, который уже присутствует в словаре d, выполняется замена соответствующим значением из a; a может быть словарем, итерируемым объектом с парами (ключ, значение) или именованными аргументами
d.fromkeys(s [, v])	Возвращает словарь типа dict, ключами которого являются элементы последовательности s, а значениями либо None, либо v, если аргумент v определен



ИСПОЛЬЗОВАНИЕ СЛОВАРНЫХ МЕТОДОВ

Примеры скриптов:

- Инициирование словарей и работу с его значениями смотрите в [dict_init.py](#)
- Манипуляции со значениями в словарях смотрите в [dict_values.py](#)



СЛОВАРИ СО ЗНАЧЕНИЯМИ ПО УМОЛЧАНИЮ

- Иницируются с помощью `collections.defaultdict()`
- Поддерживают все методы обычных словарей
- Не выдают исключение `KeyError` при обращении к отсутствующему элементу
- См. примеры в `unique_words_1.py` и `unique_words_2.py`



1. Какой словарь создаст этот код:
`>>> {}.fromkeys(['name', 'age'])`
2. А этот:
`>>> {}.fromkeys(['name', 'age'], 123)`
3. Что здесь ключи, а что значения:
`>>> key_val = {'a': 1, 2: 'b'}`
4. Какой результат вернет выражение
`>>> key_val[2]`

???



ПРАКТИЧЕСКОЕ ЗАДАНИЕ.

ОПРЕДЕЛЕНИЕ ЧАСТОТЫ ВСТРЕЧАЕМОСТИ СЛОВ

Когда Лев Толстой написал «Войну и мир», ему стало интересно, какие слова и в каком количестве используются в его романе.

Напишите программу, которая подсчитывает и выводит частоту встречаемости слов в тексте.

- Считайте текст из файла *Data/War & Peace.txt*
- Избавьтесь от знаков пунктуации: . , ! ? : – и символов перевода строки
- Посчитайте для каждого уникального слова число его повторений (без учёта регистра) и занесите результат в словарь, где **ключ – слово, значение – частота** его встречаемости в тексте
- Запишите результат в файл *out.txt* в формате «слово : частота» в порядке убывания частоты встречаемости слов



ПРАКТИЧЕСКОЕ ЗАДАНИЕ.

ОБМЕН КЛЮЧЕЙ СО ЗНАЧЕНИЯМИ

Есть такой словарь:

```
>>> fruit_2_color = {'watermelon': 'green', 'pomegranate': 'red',  
'peach': 'orange', 'cherry': 'red', 'pear': 'green',  
'banana': 'yellow', 'plum': 'purple', 'orange': 'orange'}
```

Код для обмена key и value:

```
>>> color_2_fruit = {}  
>>> for fruit in fruit_2_color:  
    color = fruit_2_color[fruit]  
    color_2_fruit[color] = fruit
```

- Что сделает с данными этот код?
- Как это исправить?



ПРАКТИЧЕСКОЕ ЗАНЯТИЕ.

СЛОВАРЬ ПРОГРАММИСТА

- План программы:
 - Описать слова из жаргона программиста
 - Предложить пользователю меню:
 - Поиск термина
 - Добавление термина
 - Обновление толкования термина
 - Удаление термина
 - Вывод терминов по алфавиту
- См. программу в [proger_translator.py](#)



ПРАКТИЧЕСКОЕ ЗАНЯТИЕ.

ФОРМАТ JSON

- JSON = Java Script Object Notation
- Это широко распространённый формат хранения и передачи информации
- Формат JSON совпадает с литеральным определением словаря в Python
- Смотрите пример файла в [EnRu/dict.json](#)
- См. программу в [EnRu/json_open.py](#)



ПРАКТИЧЕСКОЕ ЗАНЯТИЕ.

ПОИСК В АНГЛО-РУССКОМ СЛОВАРЕ

- Запрашиваем слово на английском
- Ищем это слово по ключу в словаре – смотрите функцию `translate()` в `EnRu/tran_start.py`
- Добавляем проверку – присутствует ли такое слово среди ключей словаря
- Тем самым избавляемся от появления исключения `KeyError`
- Пример в `EnRu/tran_no_exc.py`



ПРАКТИЧЕСКОЕ ЗАДАНИЕ.

ПОИСК БЕЗ УЧЁТА РЕГИСТРА

- Большинство слов в нижнем регистре
- Преобразуем искомое слово к нижнему регистру – см. [EnRu/tran_any_case.py](#)
- Как быть с поиском:
 - слов с большой буквы – “Friday”, “December”, “Magnitogorsk”, ... ?
 - слов крупными буквами – “OK”, “TV”, ... ?
- **Доработайте программу для обработки таких ситуаций**



ПРАКТИЧЕСКОЕ ЗАНЯТИЕ.

БИБЛИОТЕКА **DIFFLIB**

- **difflib** – стандартная библиотека Python
- Используемые типы и функции **difflib**:
 - **SequenceMatcher** – тип данных для выполнения сравнений
 - **ratio()** – метод-функция объекта типа **SequenceMatcher**, выдающая долю совпадений
 - **get_close_matches()** – функция поиска ближайших совпадений в списке
- Смотрите пример в **difflib_samples.py**



ПРАКТИЧЕСКОЕ ЗАДАНИЕ.

ПРЕДЛОЖЕНИЕ ПОХОЖИХ СЛОВ

- Если слово не найдено, нужно предложить похожее
- Если есть похожее – спросить пользователя его ли он имел ввиду
- Используйте библиотеку `difflib`
- Доработайте программу «Англо-русский словарь» для предложения похожих слов



ДОМАШНЕЕ ЗАДАНИЕ.

АНГЛО-РУССКИЙ СЛОВАРЬ V.2.0

- Добавьте меню – первые 4 пункта как в «Словаре программиста»:
 - 0 - Выйти
 - 1 - Найти толкование термина
 - 2 - Добавить термин
 - 3 - Изменить толкование термина
 - 4 - Удалить термин
- Добавьте ещё один пункт:
 - 5 - Сохранить в файл
- Разберитесь как сохранять данные из словаря в файл с помощью модуля `json`

СПАСИБО ЗА ВНИМАНИЕ !
ВОПРОСЫ ?



*School of
Computer
Science*