Лабораторная работа №11
Словари

1 Цель работы

Изучить словари и научиться применять полученные знания на практике.

2 Краткая теория

2.1 Словари

Обычные списки (массивы) представляют собой набор пронумерованных элементов, то есть для обращения к какому-либо элементу списка необходимо указать его номер. Номер элемента в списке однозначно идентифицирует сам элемент. Но идентифицировать данные по числовым номерам не всегда оказывается удобно. Например, маршруты поездов в России идентифицируются численно-буквенным кодом (число и одна буква), также численно-буквенным кодом идентифицируются авиарейсы, то есть для хранения информации о рейсах поездов или самолетов в качестве идентификатора удобно было бы использовать не число, а текстовую строку.

Структура данных, позволяющая идентифицировать ее элементы не по числовому индексу, а по произвольному, называется словарем или ассоциативным массивом. Соответствующая структура данных в языке Питон называется dict.

Рассмотрим простой пример использования словаря. Заведем словарь Capitals, где индексом является название страны, а значением – название столицы этой страны. Это позволит легко определять по строке с названием страны ее столицу.

```
# Создадим пустой словарь Capitals
Capitals = dict()

# Заполним его несколькими значениями
Capitals['Russia'] = 'Moscow'
Capitals['Ukraine'] = 'Kiev'
Capitals['USA'] = 'Washington'

Countries = ['Russia', 'France', 'USA', 'Russia']

for country in Countries:
    # Для каждой страны из списка проверим, есть ли она в словаре Capitals
    if country in Capitals:
        print('Столица страны ' + country + ': ' + Capitals[country])
    else:
        print('В базе нет страны с названием ' + country)
```

Итак, каждый элемент словаря состоит из двух объектов: ключа и значения. В нашем примере ключом является название страны, значением является название столицы. Ключ идентифицирует элемент словаря, значение является данными, которые соответствуют данному ключу. Значения ключей – уникальны, двух одинаковых ключей в словаре быть не может.

В жизни широко распространены словари, например, привычные бумажные словари (толковые, орфографические, лингвистические). В них ключом является слово-заголовок статьи, а значением – сама статья. Для того, чтобы получить доступ к статье, необходимо указать слово-ключ.

Другой пример словаря, как структуры данных – телефонный справочник. В нем ключом является имя, а значением – номер телефона. И словарь, и телефонный справочник хранятся так, что легко найти элемент словаря по известному ключу (например, если записи хранятся в алфавитном порядке ключей, то легко можно найти известный ключ, например, бинарным поиском), но если ключ неизвестен, а известно лишь значение, то поиск элемента с данным значением может потребовать последовательного просмотра всех элементов словаря.

Особенностью ассоциативного массива является его динамичность: в него можно добавлять новые элементы с произвольными ключами и удалять уже существующие элементы. При этом размер используемой памяти пропорционален размеру ассоциативного массива. Доступ к элементам ассоциативного массива выполняется хоть и медленнее, чем к обычным массивам, но в целом довольно быстро.

В языке Питон ключом может быть произвольный неизменяемый тип данных: целые и действительные числа, строки, кортежи. Ключом в словаре не может быть множество, но может быть элемент типа frozenset: специальный тип данных, являющийся аналогом типа set, который нельзя изменять после создания. Значением элемента словаря может быть любой тип данных, в том числе и изменяемый.

2.2 Когда нужно использовать словари

Словари нужно использовать в следующих случаях:

1. Подсчет числа каких-то объектов. В этом случае нужно завести словарь, в котором ключами являются объекты, а значениями – их количество.
2. Хранение каких-либо данных, связанных с объектом. Ключи – объекты, значения – связанные с ними данные. Например, если нужно по названию месяца определить его порядковый номер, то это можно сделать при помощи словаря Num['January'] = 1; Num['February'] = 2; ….
3. Установка соответствия между объектами (например, «родитель-потомок»). Ключ – объект, значение – соответствующий ему объект.
4. Если нужен обычный массив, но максимальное значение индекса элемента очень велико, и при этом будут использоваться не все возможные индексы (так называемый «разреженный массив»), то можно использовать ассоциативный массив для экономии памяти.

2.3 Создание словаря

Пустой словарь можно создать при помощи функции dict() или пустой пары фигурных скобок {} (вот почему фигурные скобки нельзя использовать для создания пустого множества). Для создания словаря с некоторым набором начальных значений можно использовать следующие конструкции:

```
Capitals = {'Russia': 'Moscow', 'Ukraine': 'Kiev', 'USA': 'Washington'}
Capitals = dict(Russia = 'Moscow', Ukraine = 'Kiev', USA = 'Washington')
Capitals = dict([("Russia", "Moscow"), ("Ukraine", "Kiev"), ("USA",
"Washington")])
Capitals = dict(zip(["Russia", "Ukraine", "USA"], ["Moscow", "Kiev",
"Washington"]))
print(Capitals)
```

Первые два способа можно использовать только для создания небольших словарей, перечисляя все их элементы. Кроме того, во втором способе ключи передаются как именованные параметры функции dict, поэтому в этом случае ключи могут быть только строками, причем являющимися корректными идентификаторами. В третьем и четвертом случае можно создавать большие словари, если в качестве аргументов передавать уже готовые списки, которые могут быть получены не обязательно перечислением всех элементов, а любым другим способом построены по ходу исполнения программы. В третьем способе функции dict нужно передать список, каждый элемент которого является кортежем из двух элементов: ключа и значения. В четвертом способе используется функция zip, которой передаются два списка одинаковой длины: список ключей и список значений.

2.4 Работа с элементами словаря

Основная операция: получение значения элемента по ключу, записывается так же, как и для списков: A[key]. Если элемента с заданным ключом нет в словаре, то возникает исключение KeyError.

Другой способ определения значения по ключу – метод get: A.get(key). Если элемента с ключом get нет в словаре, то возвращается значение None. В форме записи с двумя аргументами A.get(key, val) метод возвращает значение val, если элемент с ключом key отсутствует в словаре.

Проверить принадлежность элемента словарю можно операциями in и not in, как и для множеств.

Для добавления нового элемента в словарь нужно просто присвоить ему какое-то значение: A[key] = value.

Для удаления элемента из словаря можно использовать операцию del A[key] (операция возбуждает исключение KeyError, если такого ключа в словаре нет. Вот два безопасных способа удаления элемента из словаря.

```
A = {'ab' : 'ba', 'aa' : 'aa', 'bb' : 'bb', 'ba' : 'ab'}

key = 'ac'
if key in A:
    del A[key]
try:
    del A[key]
except KeyError:
   print('There is no element with key "' + key + '" in dict')
print(A)
```

В первом случае мы предварительно проверяем наличие элемента, а во втором – перехватываем и обрабатываем исключение.

Еще один способ удалить элемент из словаря: использование метода pop: A.pop(key). Этот метод возвращает значение удаляемого элемента, если

элемент с данным ключом отсутствует в словаре, то возбуждается исключение. Если методу pop передать второй параметр, то если элемент в словаре отсутствует, то метод pop возвратит значение этого параметра. Это позволяет проще всего организовать безопасное удаление элемента из словаря: A.pop(key, None).

2.5 Перебор элементов словаря

Можно легко организовать перебор ключей всех элементов в словаре:

```
A = dict(zip('abcdef', list(range(6))))
for key in A:
    print(key, A[key])
```

Следующие методы возвращают представления элементов словаря. Представления во многом похожи на множества, но они изменяются, если менять значения элементов словаря. Метод keys возвращает представление ключей всех элементов, метод values возвращает представление всех значений, а метод items возвращает представление всех пар (кортежей) из ключей и значений.

Соответственно, быстро проверить, есть ли значение val среди всех значений элементов словаря A можно так: val in A.values(), а организовать цикл так, чтобы в переменной key был ключ элемента, а в переменной val, было его значение можно так:

```
A = dict(zip('abcdef', list(range(6))))
for key, val in A.items():
    print(key, val)
```

3 Порядок выполнения работы

Получить задание для выполнения лабораторной работы (раздел 4) согласно своему варианту (по журналу). Разработать программу.

4 Задания для выполнения работы

Задание 1. Номер появления слова.

В единственной строке записан текст. Для каждого слова из данного текста подсчитайте, сколько раз оно встречалось в этом тексте ранее.

Словом считается последовательность непробельных символов идущих подряд, слова разделены одним или большим числом пробелов или символами конца строки.

Входные данные	Правильный ответ
one two one tho three	0 0 1 0 0

Задание 2. Словарь синонимов.

Вам дан словарь, состоящий из пар слов. Каждое слово является синонимом к парному ему слову. Все слова в словаре различны.

Для слова из словаря, записанного в последней строке, определите его синоним.

Входные данные	Правильный ответ
3 Hello Hi Bye Goodbye	Bye

List Array Goodbye	

Задание 3. Выборы в США. Как известно, в США президент выбирается не прямым голосованием, а путем двухуровневого голосования. Сначала проводятся выборы в каждом штате и определяется победитель выборов в данном штате. Затем проводятся государственные выборы: на этих выборах каждый штат имеет определенное число голосов – число выборщиков от этого штата. На практике, все выборщики от штата голосуют в соответствии с результатами голосования внутри штата, то есть на заключительной стадии выборов в голосовании участвуют штаты, имеющие различное число голосов.

В первой строке дано количество записей. Далее, каждая запись содержит фамилию кандидата и число голосов, отданных за него в одном из штатов. Подведите итоги выборов: для каждого из участника голосования определите число отданных за него голосов. Участников нужно выводить в алфавитном порядке.

Входные данные	Правильный ответ
5 McCain 10 McCain 5 Obama 9 Obama 8 McCain 1	McCain 16 Obama 17

Задание 4. Самое частое слово.

Дан текст: в первой строке задано число строк, далее идут сами строки. Выведите слово, которое в этом тексте встречается чаще всего. Если таких слов несколько, выведите то, которое меньше в лексикографическом порядке.

Входные данные	Правильный ответ
1 apple orange banana banana orange	banana

Задание 5. Права доступа.

В файловую систему одного суперкомпьютера проник вирус, который сломал контроль за правами доступа к файлам. Для каждого файла известно, с какими действиями можно к нему обращаться:

1) запись W;

2) чтение R;

3) запуск X.

В первой строке содержится число N – количество файлов содержащихся в данной файловой системе. В следующих N строчках содержатся имена файлов и допустимых с ними операций, разделенные пробелами. Далее указано число M – количество запросов к файлам. В последних M строках указан запрос вида Операция Файл. К одному и тому же файлу может быть применено любое количество запросов.

Вам требуется восстановить контроль над правами доступа к файлам (ваша программа для каждого запроса должна будет возвращать OK если над файлом выполняется допустимая операция, или же Access denied, если операция недопустима.

Входные данные	Правильный ответ
4 helloworld.exe R X pinglog W R nya R goodluck X W R 5 read nya write helloworld.exe execute nya read pinglog write pinglog	OK Access denied Access denied OK OK

Задание 6. Частотный анализ.

Дан текст: в первой строке записано количество строк в тексте, а затем сами строки. Выведите все слова, встречающиеся в тексте, по одному на каждую строку. Слова должны быть отсортированы по убыванию их количества появления в тексте, а при одинаковой частоте появления – в лексикографическом порядке.

Указание. После того, как вы создадите словарь всех слов, вам захочется отсортировать его по частоте встречаемости слова. Желаемого можно добиться, если создать список, элементами которого будут кортежи из двух элементов: частота встречаемости слова и само слово. Например, [(2, 'hi'), (1, 'what'), (3, 'is')]. Тогда стандартная сортировка будет сортировать список кортежей, при этом кортежи сравниваются по первому элементу, а если они равны – то по второму. Это почти то, что требуется в задаче.

Входные данные	Правильный ответ
9 hi hi what is your name my name is bond james bond my name is damme van damme claude van damme jean claude van damme	damme is name van bond claude hi my james jean what your

Задание 7. Страны и города.

Дан список стран и городов каждой страны. Затем даны названия городов. Для каждого города укажите, в какой стране он находится.

Входные данные	Правильный ответ
2 Russia Moscow Petersburg Novgorod Kaluga Ukraine Kiev Donetsk Odessa 3 Odessa Moscow Novgorod	Ukraine Russia Russia

Задание 8. Англо-латинский словарь.

Однажды, разбирая старые книги на чердаке, школьник Вася нашёл англо-латинский словарь. Английский он к тому времени знал в совершенстве, и его мечтой было изучить латынь. Поэтому попавшийся словарь был как раз кстати.

К сожалению, для полноценного изучения языка недостаточно только одного словаря: кроме англо-латинского необходим латинско-английский. За неимением лучшего он решил сделать второй словарь из первого.

Как известно, словарь состоит из переводимых слов, к каждому из которых приводится несколько слов-переводов. Для каждого латинского слова, встречающегося где-либо в словаре, Вася предлагает найти все его переводы (то есть все английские слова, для которых наше латинское встречалось в его списке переводов), и считать их и только их переводами этого латинского слова.

Помогите Васе выполнить работу по созданию латинско-английского словаря из англо-латинского.

В первой строке содержится единственное целое число N – количество английских слов в словаре. Далее следует N описаний. Каждое описание содержится в отдельной строке, в которой записано сначала английское слово, затем отделённый пробелами дефис, затем разделённые запятыми с пробелами переводы этого английского слова на латинский. Все слова состоят только из маленьких латинских букв. Переводы отсортированы в лексикографическом порядке. Порядок следования английских слов в словаре также лексикографический.

Выведите соответствующий данному латинско-английский словарь, в точности соблюдая формат входных данных. В частности, первым должен идти перевод лексикографически минимального латинского слова, далее – второго в этом порядке и т.д. Внутри перевода английские слова должны быть также отсортированы лексикографически.

Входные данные	Правильный ответ
3 apple - malum, pomum, popula fruit - baca, bacca, popum punishment - malum, multa	7 baca - fruit bacca - fruit malum - apple, punishment multa - punishment pomum - apple

	popula - apple popum - fruit

Задание 9. Контрольная по ударениям.

Учительница задала Пете домашнее задание – в заданном тексте расставить ударения в словах, после чего поручила Васе проверить это домашнее задание. Вася очень плохо знаком с данной темой, поэтому он нашел словарь, в котором указано, как ставятся ударения в словах. К сожалению, в этом словаре присутствуют не все слова. Вася решил, что в словах, которых нет в словаре, он будет считать, что Петя поставил ударения правильно, если в этом слове Петей поставлено ровно одно ударение.

Оказалось, что в некоторых словах ударение может быть поставлено больше, чем одним способом. Вася решил, что в этом случае если то, как Петя поставил ударение, соответствует одному из приведенных в словаре вариантов, он будет засчитывать это как правильную расстановку ударения, а если не соответствует, то как ошибку.

Вам дан словарь, которым пользовался Вася и домашнее задание, сданное Петей. Ваша задача – определить количество ошибок, которое в этом задании насчитает Вася.

Вводится сначала число N – количество слов в словаре.

Далее идет N строк со словами из словаря. Каждое слово состоит не более чем из 30 символов. Все слова состоят из маленьких и заглавных латинских букв. В каждом слове заглавная ровно одна буква – та, на которую попадает ударение. Слова в словаре расположены в алфавитном порядке. Если есть несколько возможностей расстановки ударения в одном и том же слове, то эти варианты в словаре идут в произвольном порядке.

Далее идет упражнение, выполненное Петей. Упражнение представляет собой строку текста, суммарным объемом не более 300000 символов. Строка состоит из слов, которые разделяются между собой ровно одним пробелом. Длина каждого слова не превышает 30 символов. Все слова состоят из маленьких и заглавных латинских букв (заглавными обозначены те буквы, над которыми Петя поставил ударение). Петя мог по ошибке в каком-то слове поставить более одного ударения или не поставить ударения вовсе.

Выведите количество ошибок в Петином тексте, которые найдет Вася.

Входные данные	Правильный ответ
4 cAnnot cannOt fOund pAge thE pAge cAnnot be found	2

Задание 10. Продажи.

Дана база данных о продажах некоторого интернет-магазина. Каждая строка входного файла представляет собой запись вида Покупатель товар количество, где Покупатель – имя покупателя (строка без пробелов),

товар – название товара (строка без пробелов), количество – количество приобретенных единиц товара.

Создайте список всех покупателей, а для каждого покупателя подсчитайте количество приобретенных им единиц каждого вида товаров. Список покупателей, а также список товаров для каждого покупателя нужно выводить в лексикографическом порядке.

Входные данные	Правильный ответ
Ivanov paper 10 Petrov pens 5 Ivanov marker 3 Ivanov paper 7 Petrov envelope 20 Ivanov envelope 5	Ivanov: envelope 5 marker 3 paper 17 Petrov: envelope 20 pens 5

Задание 11. Родословная: подсчет уровней.

В генеалогическом древе у каждого человека, кроме родоначальника, есть ровно один родитель.

Каждом элементу дерева сопоставляется целое неотрицательное число, называемое высотой. У родоначальника высота равна 0, у любого другого элемента высота на 1 больше, чем у его родителя.

Вам дано генеалогическое древо, определите высоту всех его элементов.

Программа получает на вход число элементов в генеалогическом древе N. Далее следует $N-1$ строка, задающие родителя для каждого элемента древа, кроме родоначальника. Каждая строка имеет вид имя_потомка имя_родителя.

Программа должна вывести список всех элементов древа в лексикографическом порядке. После вывода имени каждого элемента необходимо вывести его высоту.

Примечание

Эта задача имеет решение сложности $O(n)$, но вам достаточно написать решение сложности $O(n^2)$ (не считая сложности обращения к элементам словаря).

Входные данные	Правильный ответ
9 Alexei Peter_I Anna Peter_I Elizabeth Peter_I Peter_II Alexei Peter_III Anna Paul_I Peter_III Alexander_I Paul_I Nicholaus_I Paul_I	Alexander_I 4 Alexei 1 Anna 1 Elizabeth 1 Nicholaus_I 4 Paul_I 3 Peter_I 0 Peter_II 2 Peter_III 2

Задание 12. Родословная: предки и потомки.

Даны два элемента в дереве. Определите, является ли один из них потомком другого.

Во входных данных записано дерево в том же формате, что и в предыдущей задаче Далее идет число запросов K. В каждой из следующих K строк, содержатся имена двух элементов дерева.

Для каждого такого запроса выведите одно из трех чисел: 1, если первый элемент является предком второго, 2, если второй является предком первого или 0, если ни один из них не является предком другого.

Входные данные	Правильный ответ
9 Alexei Peter_I Anna Peter_I Elizabeth Peter_I Peter_II Alexei Peter_III Anna Paul_I Peter_III Alexander_I Paul_I Nicholaus_I Paul_I 3 Anna Nicholaus_I Peter_II Peter_I Alexei Paul_I	1 2 0

Задание 13. Родословная: LCA.

В генеалогическом древе определите для двух элементов их наименьшего общего предка (Lowest Common Ancestor). Наименьшим общим предком элементов A и B является такой элемент C, что C является предком A, C является предком B, при этом глубина C является наибольшей из возможных. При этом элемент считается своим собственным предком.

Формат входных данных аналогичен предыдущей задаче.

Для каждого запроса выведите наименьшего общего предка данных элементов.

Входные данные	Правильный ответ
9 Alexei Peter_I Anna Peter_I Elizabeth Peter_I Peter_II Alexei Peter_III Anna Paul_I Peter_III Alexander_I Paul_I Nicholaus_I Paul_I 3 Alexander_I Nicholaus_I Peter_II Paul_I	Paul_I Peter_I Anna

Alexander_I Anna	