

Министерство науки и высшего образования Российской Федерации

Рязанский государственный радиотехнический университет
имени В.Ф.Уткина

Python.

Словари. Создание и простейшая обработка

Методические указания к лабораторной работе

19



Рязань 2022

УДК 004.432

Python. Словари. Создание и простейшая обработка: Методические указания к лабораторной работе № 18 / Рязан. гос. радиотехн. универ.; сост.: А.П. Кирсанов, А.В. Климухина, А.Н. Пылькин Ю.С.Соколова, Е.С. Щенёв, М.Г. Щетинин. – Рязань, 2022. – 17 с.

Рассмотрены вопросы обработки информации, содержащейся в наиболее часто и эффективно используемых последовательностях в языке Python – словарях (тип данных `dictionary`), реализующих структуру данных в форме неупорядоченных последовательностей формата «ключ: значение». В первую очередь внимание уделено наиболее часто используемым подходам организации и обработки данных в списках: создание словаря, получение данных из словаря, добавление элементов и пр.. Определены правила работы с основными методами, определенными для класса `dict`.

В качестве практических заданий предлагается составить программу в которой реализуется создание и простейшая обработка списков.

Предназначено для студентов направлений 09.03.03 – Прикладная информатика и 09.03.04 – Программная инженерия по дисциплине «Алгоритмические языки и программирование», а также для студентов очной и заочной формы обучения всех направлений подготовки и специальностей.

Коллекции, списки, создание списка, обработка данных в списке, методы работы со списками.

Печатается по решению Научно-методического совета Рязанского государственного радиотехнического университета имени В.Ф. Уткина.

Рецензент: кафедра информатики, информационных технологий и защиты информации ФГБОУ ВО «Липецкий государственный педагогический университет им. П.П. Семенова-Тян-Шанского» (зав. каф., к.т.н., доц. Скуднев Д.М.)

Составители: Кирсанов Александр Павлович
Климухина Анастасия Витальевна
Пылькин Александр Николаевич
Соколова Юлия Сергеевна
Щенёв Евгений Сергеевич
Щетинин Максим Геннадьевич

Словари. Создание и простейшая обработка.

Среди различных встроенных структур данных, используемых для хранения разных типов информации, в Python наиболее часто используется важная структура словарь (dict). Словарь обеспечивает хранение и обработку данных в виде неупорядоченной последовательности по формату пар «ключ: значения», что отличает его от обычных списков (массивов), в которых обращение к элементу реализуется по номеру (индексу) этого элемента в массиве. В словаре идентификация элементов осуществляется по ключу, что в ряде приложений становится весьма удобным приемом. В описании языка Python наряду с термином «словарь» используются и другие названия:

- ассоциативные массивы (associative arrays);
- ассоциативные хэш-таблицы (hashmaps);
- поисковые таблицы (lookup tables);
- таблицы преобразования.

Наиболее наглядным примером словаря можно считать справочную телефонную книгу, в которой хранятся справочная информация в формате «фамилия: номер телефона». Пусть, например, телефонная книга (в виде словаря) содержит сведения о телефонах трех человек:

```
telephone = {'иванов': '910-555-66-77',  
            'петров': '920-888-11-23',  
            'сидоров': '915-555-23-45'}  
# обращение к элементу словаря с помощью ключа  
print(telephone['петров'])  
# или такое обращение  
x = telephone['петров']  
print(x)  
# в обоих случаях выводится номер 920-888-11-23
```

Данный словарь telephone в качестве ключей используются строки, в которых указаны фамилии обладателей телефонов. Ключом может быть любой неизменяемый тип данных, а значением конкретного ключа может быть что угодно.

Если в качестве ключа используются числа, то примером может служить словарь, сформированный по правилам использования номера

факультета, которые приняты в Рязанском государственном радиотехническом университете им. В.Ф. Уткина. В качестве значений в словаре используются строки.

```
facultet = {1: 'ФРТ', 2: 'ФЭ', 3: 'ФАИТУ', 4: 'ФВТ',  
5: 'ИЭФ'}  
print(facultet)  
# создан и выводится словарь  
# {1: 'ФРТ', 2: 'ФЭ', 3: 'ФАИТУ', 4: 'ФВТ', 5: 'ИЭФ'}
```

Таким образом, ключом может быть любой неизменяемый тип данных (целые числа, действительные числа, строки и кортежи). В то же время в качестве ключа нельзя использовать списки и множества. При этом допускается использование типа frozenset – специальный тип данных, который является аналогом типа set и который нельзя изменять после его создания. Значением словаря может быть любой тип (в том числе изменяемый). Если использовать изменяемый тип данных в качестве ключа, то это приведет к ошибке, как это происходит в следующем примере:

```
dictionary = {1: 'целые числа могут быть ключами',  
              (1, 2.0): 'кортежи могут быть ключами',  
              'иванов': 'строки могут быть ключами',  
              ['один', 2, 3]: 'списки не могут быть ключами'}
```

Создание словаря

Первая простейшая операция, с которой сталкивается программист при использовании типа dict, является создание словаря, как некоторой коллекции данных. При создании словаря необходимо передать (указать) последовательность элементов внутри фигурных скобок {}, разделив их запятыми. Каждый элемент имеет ключ и значение. Значения могут представлять собой любой тип данных и могут повторяться, но ключи, которые обеспечивают доступ к тому или иному значению, должны быть уникальными. Существует несколько способов создания словарей.

Способ 1. Использование литерала (литеральной коллекции)

Напомним, что в программировании литералом называют выражение или константу, которое создает некоторый объект определенного типа.

```
d = {}      # создание пустого словаря d
facultet = {1: 'ФРТ', 2: 'ФЭ', 3: 'ФАИТУ', 4: 'ФВТ',
            5: 'ИЭФ', 'институт': 'ИМИА'}
#         создание словаря facultet с ключами разных типов
#         {1: 'ФРТ', 2: 'ФЭ', 3: 'ФАИТУ',
#         4: 'ФВТ', 5: 'ИЭФ', 'институт': 'ИМИА'}
```

Способ 2. С помощью метода dict().

Данный способ использует свойства последовательностей в языке Python. Примеры использования функции dict:

```
d1 = dict(short='dict', long='dictionary',
abc_21=4567)
print(d1)
d2 = dict([(1, 1), ('2', (3, 4, 'пять'))], ("три",
"семь"))
print(d2)
d3 = dict({1: 'A', 2: (1, 2, 'три'), 3: 4.56789})
print(d3)
```

Формируются словари с именами d1, d2 и d3:

```
{'short': 'dict', 'long': 'dictionary', 'abc_21': 4567}
```

```
{1: 1, '2': (3, 4, 'пять'), 'три': 'семь'}
```

```
{1: 'A', 2: (1, 2, 'три'), 3: 4.56789}
```

Способ 3. Использование генератора словарей.

Генераторы словарей очень похожи на генераторы списков. Напоминаем, что генерация осуществляется с нулевого значения.

```
dict_generator = {a: a**3 for a in range(5)}  
print(dict_generator)  
# формирование и вывод словарь dict_generator  
# {0: 0, 1: 1, 2: 8, 3: 27, 4: 64}
```

Один словарь может входить в состав другого словаря. В результате получается вложенный словарь (комплексный словарь):

```
session = {  
    'фамилия': {1: "иванов",  
                2: 'сидоров',  
                3: 'петров'},  
    'дисциплина': {1: 'математика',  
                   2: 'физика',  
                   3: 'алгоритмические языки и  
программирование',  
                   4: 'история'}}}
```

Получение данных из словаря

Для обращения к элементу списка (массива) указывается номер элемента, к которому осуществляется обращение. Аналогичным образом осуществляется доступ к элементу словаря, при этом в квадратных скобках указывается ключ (вместо номера элемента). Во избежание ошибок необходимо указывать только существующий ключ. В приведенном ниже примере показаны разные способы обращения к элементам словаря (в том числе для случая вложенного словаря). Обращение к данным реализовано для рассмотренных ранее словарей `dict.generator`, `facultet` и `session`.

```
# формирование словаря dict_generator  
# {0: 0, 1: 1, 2: 8, 3: 27, 4: 64}  
dict_generator = {a: a**3 for a in range(5)}  
# обращение к четвертому элементу  
x = dict_generator[3]  
print('число 3 в кубе равно = ', x)  
# формирование словаря facultet
```

```

facultet = {1: 'ФРТ', 2: 'ФЭ', 3: 'ФАИТУ', 4: 'ФВТ',
            5: 'ИЭФ', 'институт': 'ИМИА'}
# обращение и вывод четвертого элемента
print('факультет - ', facultet[4])
# формирование вложенного словаря и вывод элемента
session = {
    'фамилия': {1: "иванов",
                2: 'сидоров',
                3: 'петров'},
    'дисциплина': {1: 'математика',
                  2: 'физика',
                  3: 'алгоритмические языки и
программирование',
                  4: 'история'}}
print(session['дисциплина'][3])

print(session['дисциплина'][3])

```

Результат выполнения программы:

```

число 3 в кубе равно = 27
факультет - ФВТ
алгоритмические языки и программирование

```

Использование квадратных скобок при обращении к данным словаря считается «не слишком хорошим» способом. Если указанного в квадратных скобках ключа нет, то идентифицируется исключение KeyError. Эта ситуация считается неприятной, особенно в тех случаях, когда обрабатываются большие данные с частым изменением.

Для исключения этой ситуации можно проверить наличие указанного ключа в словаре с помощью выражения «ключ in словарь»:

```

users = {
    '11111111': 'Ваня',
    '22222222': 'Коля',
    '44444444': 'маша'
}

```

```
key = '33333333'
if key in users:
    user = users[key]
    print(user)
else:
    print('элемент не найден')
```

Более предпочтительный способ обращения к значениям словаря считается использование функции `get()`, определенной над классом `dict`. Ниже приведена программа, в которой реализованы вызовы значений из словаря посредством функции `.get()`.

```
# формирование словаря dict_generator
# {0: 0, 1: 1, 2: 8, 3: 27, 4: 64}
dict_generator = {a: a**3 for a in range(5)}
# обращение к четвертому элементу
x = dict_generator[3]
print('число 3 в кубе равно = ', x)
# формирование словаря facultet
facultet = {1: 'ФРТ', 2: 'ФЭ', 3: 'ФАИТУ', 4: 'ФВТ',
            5: 'ИЭФ', 'институт': 'ИМИА'}
# обращение и вывод четвертого элемента
print('факультет - ', facultet[4])
# формирование вложенного словаря и вывод элемента
session = {
    'фамилия': {1: "иванов",
                2: 'сидоров',
                3: 'петров'},
    'дисциплина': {1: 'математика',
                   2: 'физика',
                   3: 'алгоритмические языки и
программирование',
                   4: 'история'}}
#
x = dict_generator.get(3)
print('число 3 в кубе равно = ', x)
#
print('факультет - ', facultet.get(4))
#
print(session.get('фамилия'))
```



```
print(session.get('дисциплина'[2]))
```

Результаты выполнения программы:

число 3 в кубе равно = 27

факультет - ФВТ

{1: 'иванов', 2: 'сидоров', 3: 'петров'}

None

Иногда применяется метод `.setdefault()`, который синтаксически похож на использование `.get()`, использующий два параметра: первый – вызываемый ключ, второй – значение ключа по умолчанию (`None`). Если указывается второй параметр в методе `.setdefault()`, то создается еще пара «ключ-значение» (см. далее при описании других методов, используемых при обработке словарей).

Добавление элементов

Существуют разные способы добавление новых элементов в словарь. Ниже приведен программный код, демонстрирующий различные способы. В начале создается пустой словарь `D`, в который помещается элемент «пн»:1, где «пн»-ключ, 1 – значение. Далее в словарь добавляются два элемента один за другим. Отдельно указываются ключи и соответствующие значения. Последующий текст программы показывает примеры добавления нескольких значений для одного ключа.

```
# для компактного написания кода использованы
# сокращения: пн - понедельник, чт - четверг,
# пт - пятница, сб - суббота, вс - воскресенье
# создание пустого словаря
D = {}
print(D)
# добавление элемента с ключом 'пн'
D['пн'] = 1
```

```

print(D)
# добавление двух элементов
D['чт'] = 4
D['пт'] = 5
print(D)
# добавление нескольких значений
# для одного ключа
D['вых'] = 'сб', 'вс'
print(D)
D['день_недели'] = (1, 2, 3, 4, 5, 6, 'вс')
print(D)

```

Результат работы операторов print:

```

{}
{'пн': 1}
{'пн': 1, 'чт': 4, 'пт': 5}
{'пн': 1, 'чт': 4, 'пт': 5, 'вых': ('сб', 'вс')}
{'пн': 1, 'чт': 4, 'пт': 5, 'вых': ('сб', 'вс'),
 'день_недели': (1, 2, 3, 4, 5, 6, 'вс')}

```

Удаление элементов

Приведенный ниже программный код демонстрирует приемы удаления элементов словаря. В качестве исходного взят словарь D, сформированный в предыдущем примере (пример рассмотрен подробно ранее). Вначале из словаря D удаляется значение с ключом «вых», что реализуется с помощью использования ключевого слова `del` и указанием в квадратных скобках конкретного ключа. В результате словарь уменьшается, как это показано в комментариях. Далее для удаления пары ключ-значение используется функция `.pop()` с ключом записи в качестве аргумента. В примере удаляется запись с ключом «день недели» и словарь сокращается до трех элементов. По мере дальнейшего выполнения программы функция `.popitem()` удаляет

последний элемент словаря (указывать ключ последнего элемента не требуется). Последняя запись имела ключ «пт», что привело к ее удалению. Для удаления всех элементов в словаре можно воспользоваться функцией `clear()`, что приведет к пустому словарю `{}`.

Для удаления всего словаря используется конструкция `del<имя_словаря>`.

Программный код с поясняющими комментариями имеет вид:

```
# формирование словаря D (см. предыдущий пример)
D = {}
D['пн'] = 1
D['чт'] = 4
D['пт'] = 5
D['вых'] = 'сб', 'вс'
D['день_недели'] = (1, 2, 3, 4, 5, 6, 'вс')
print(D)
# сформированный словарь: {'пн': 1, 'чт': 4, 'пт': 5,
# 'вых': ('сб', 'вс'), 'день_недели': (1, 2, 3, 4, 5,
# 6, 'вс')}
#
# удаление элемента с ключом 'вых'
del D['вых']
print(D)
# {'пн': 1, 'чт': 4, 'пт': 5, 'день_недели': (1, 2,
# 3, 4, 5, 6, 'вс')}
# удаление элемента с ключом 'день_недели'
D.pop('день_недели')
print(D)
# {'пн': 1, 'чт': 4, 'пт': 5}
# удаление последнего элемента словаря
D.popitem()
print(D)
# {'пн': 1, 'чт': 4}
# удаление всех элементов словаря
D.clear()
print(D)
# {}
```

Копирование и объединение словарей

С целью копирования содержимого одного словаря (в ниже приведенном примере – это словарь `frieds`) в другой словарь `frieds2` используется метод `.copy()`. Объединение двух словарей `frieds` и `frieds_new` осуществляется путем использования метода `.update()`, при этом возвращается словарь из объединенных элементов.

```
frieds = {1: 'Ваня', 2: 'Коля', 4: 'Маша'}  
frieds2 = frieds.copy()  
print(frieds2)  
# {1: 'Ваня', 2: 'Коля', 4: 'Маша'}  
frieds_new = {3: 'Вася', 5: 'Таня'}  
frieds.update(frieds_new)  
print(frieds)  
# {1: 'Ваня', 2: 'Коля', 4: 'Маша', 3: 'Вася', 5: 'Таня'}
```

Перебор словаря

Перебор словаря можно организовать по разным признакам:

- перебор элементов (пар «ключ:значение»);
- перебор ключей;
- перебор значений.

Использование цикла `for` позволяет организовать перебор элементов словаря, как это реализовано в примере для словаря с именем `frieds`. Второй способ перебора базируется на применении метода `.items()`. Результаты, выводимые с помощью `print()` будут одинаковыми в первом и во втором случаях.

Для перебора ключей следует использовать метод `.keys()`, что приведет к выводу ключей в последовательности 1,2,4,3 и 5. Перебор только значений реализуется методом `.values()`, что дает возможность вывести последовательно значения «Вася», «Коля», «Маша», «Вася» и «Таня».

```

# перебор словаря
friends = {1: 'Ваня', 2: 'Коля', 4: 'Маша', 3: 'Вася', 5:
'Tаня'}
print(friends )
# использование цикла for
for num in friends:
    print(num, ' - ', friends[num])
# использование метода items()
for num, value in friends.items():
    print(num, ' - ', value)
# использование метода keys()
# перебор ключей
for num in friends.keys():
    print(num)
# использование метода values()
# перебор значений
for value in friends.values():
    print(value)
# опреление числа элементов словаря
print(len(friends))

```

Число элементов словаря можно определить с помощью метода `len()`. Последняя строки в рассматриваемом примере демонстрирует применение метода `len()`. При выводе идентифицируется значение: 5.

В заключение напомним основные методы словарей, которые представлены в виде опорных сигналов и обеспечивают закрепление рассмотренных выше теоретических сведений, связанных с обработкой информации в словарях.

Методы словарей

`dict.clear()` - очищает словарь.

`dict.copy()` - возвращает копию словаря.

classmethod **`dict.fromkeys(seq[, value])`** - создает словарь с ключами из `seq` и значением `value` (по умолчанию `None`).

`dict.get(key[, default])` - возвращает значение ключа, но если его нет, не бросает исключение, а возвращает `default` (по умолчанию `None`).

`dict.items()` - возвращает пары (ключ, значение).

dict.keys() - возвращает ключи в словаре.

dict.pop(key[, default]) - удаляет ключ и возвращает значение.

Если ключа нет, возвращает default (по умолчанию бросает исключение).

dict.popitem() - удаляет и возвращает пару (ключ, значение).

Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены.

dict.setdefault(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ со значением default (по умолчанию None).

dict.update([other]) - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!).

dict.values() - возвращает значения в словаре.

Контрольные вопросы

1. Что такое словарь и какова его структура?
2. Какие существуют способы создания словарей?
3. Перечислите способы получения данных из словаря.
4. Какие существуют способы добавления и удаления данных из словаря ?
5. Перечислите наиболее известные методы словарей.
6. Назовите методы, используемые для перечисления содержимого словаря.
7. Для чего используется метод setdefault()?
8. Каким образом можно определить число элементов словаря?

Задания

Представленные варианты заданий подразумевают составление простейшего программного кода, который содержит операции создания и элементарных преобразований данных, содержащихся в словаре. Обратите внимание на контроль содержимого создаваемых словарей и вывода значений после обработки.

Вариант 1. Даны два словаря: `dictionary_1 = {'a': 300, 'b': 400}` и `dictionary_2 = {'c': 500, 'd': 600}`. Объедините их в один словарь

dictionary_main при помощи встроенных функций языка Python, затем добавьте новый элемент с ключом 'e' и значением 1000.

Вариант 2. Даны два словаря: dictionary_1 = {'a': 500, 'b': 600} и dictionary_2 = {'c': 100, 'd': 200}. Объедините их в один словарь dictionary_main при помощи встроенных функций языка Python, затем удалите элемент с ключом 'd'.

Вариант 3. Дан словарь с числовыми значениями. Необходимо перемножить все значения, результат вывести на экран.

Вариант 4. Создайте словарь, в котором ключами будут числа от 1 до 10, а значениями эти же числа, возведенные в куб.

Вариант 5. Дан словарь с числовыми значениями. Необходимо найти максимальный элемент, вывести на экран ключ этого элемента.

Вариант 6. Дан словарь с числовыми значениями. Необходимо вычесть все значения из 1000, результат вывести на экран.

Вариант 7. Создайте словарь из строки 'python' следующим образом: в качестве ключей возьмите буквы строки, а значениями будут числа, соответствующие количеству вхождений данной буквы в строку. (подсказка: для строк можно воспользоваться функцией count())

Вариант 8. Дан словарь с числовыми значениями. Необходимо найти минимальный элемент, вывести на экран ключ этого элемента.

Вариант 9. Создайте словарь, в котором ключами будут числа от 1 до 10, а значениями эти же числа, умноженные на 2 в степени значения ключа (если ключ = 3, значит умножение на 2^3).

Вариант 10. Даны два словаря: dictionary_1 = {'a': 100, 'b': 100} и dictionary_2 = {'c': 200, 'd': 200}. Объедините их в один словарь dictionary_main при помощи встроенных функций языка Python, затем удалите элемент с ключом 'a'.

Вариант 11. Дан словарь: dictionary_1 = {'a': 1000, 'b': 400, 'c': 600, 'd': 200}. Разбейте словарь на два словаря dictionary_2 и dictionary_3 при помощи встроенных функций языка Python, затем добавьте в каждый словарь 'e': 500.

Вариант 12. Создайте словарь из 11 элементов, в котором первый элемент '1': 500, второй - '2': 400, третий - '3': 300 и т.д. Просуммируйте получившиеся элементы словаря и выведите результат.

Вариант 13. Дан словарь с числовыми значениями. Необходимо найти максимальный элемент и удалить его. Выведите на экран получившийся словарь.

Вариант 14. Дан словарь с числовыми значениями. Необходимо вывести те значения, ключи которых являются четными числами.

Вариант 15. Даны два словаря: `dictionary_1 = {'a': 700, 'b': 600}` и `dictionary_2 = {'c': 100, 'd': 200}`. Объедините их в один словарь `dictionary_main` при помощи встроенных функций языка Python, затем удалите элемент с ключом 'c'.

Вариант 16. Дан словарь с числовыми значениями. Необходимо найти среднее арифметическое значение всех элементов. Выведите результат на экран

Вариант 17. Создайте словарь, в котором ключами будут числа от 5 до 15, а значениями строки из цифры 0 в количестве, равном соответствующему ключу (ключ = 6, значение ключа = 000000).

Вариант 18. Дан словарь с числовыми значениями. Необходимо вывести те значения, ключи которых являются нечетными числами.

Вариант 19. Создайте словарь, в котором ключами будут числа от 5 до 15, а значениями эти же числа, возведенные в квадрат.

Вариант 20. Дан словарь с числовыми значениями. Необходимо суммировать все значения, результат поделить на 1000, целую часть получившегося числа вывести на экран.

Вариант 21. Дан словарь с числовыми значениями. Необходимо сложить все значения, результат вывести на экран.

Вариант 22. Создайте словарь из 11 элементов, в котором первый элемент '1': 500, а каждый последующий элемент в 5 раз меньше. Просуммируйте получившиеся элементы словаря и выведите результат.

Вариант 23. Дан словарь: `dictionary_1 = {'a': 400, 'b': 300, 'c': 200, 'd': 100}`. Разбейте словарь на два словаря `dictionary_2` и `dictionary_3` при помощи встроенных функций языка Python, затем удалите из каждого словаря по одному элементу.

Вариант 24. Необходимо создать из двух списков одинаковой длины словарь таким образом, чтобы элементы первого списка были ключами, а элементы второго — соответственно значениями нашего словаря.

Вариант 25. Необходимо создать из двух списков одинаковой длины словарь таким образом, чтобы элементы второго списка были ключами, а элементы первого — соответственно значениями нашего словаря.

Вариант 26. Дан словарь с числовыми значениями. Необходимо найти минимальный элемент и удалить его. Выведите на экран получившийся словарь.

Вариант 27. Создайте словарь из 11 элементов, в котором первый элемент '1': -500, второй - '2': -400, третий - '3': -300 и т.д. Просуммируйте получившиеся элементы словаря и выведите результат.

Вариант 28. Создайте словарь из строки 'mathematics' следующим образом: в качестве ключей возьмите буквы строки, а значениями будут числа, соответствующие количеству вхождений данной буквы в строку. (подсказка: для строк можно воспользоваться функцией `count()`)

Вариант 29. Дан словарь: `dictionary_1 = {'a': 100, 'b': 200, 'c': 300, 'd': 400}`. Разбейте словарь на два словаря `dictionary_2` и `dictionary_3` при помощи встроенных функций языка Python, затем добавьте в каждый словарь 'e': 900.

Вариант 30. Создайте словарь, в котором ключами будут числа от 1 до 10, а значениями строки из цифры 2 в количестве, равном двукратному значению соответствующего ключа (ключ = 3, значение ключа = 222222).