Словари в Пайтон

Глава 9

Пайтон для всех
www.py4e.com

Что такое Коллекция?

- Коллекция хороша тем, что мы можем положить в неё больше одного значения и хранить их все внутри одной удобной «упаковки»
- Мы можем хранить набор значений в одной переменной
- Это возможно благодаря тому, что мы имеем больше одного места «внутри» переменной
- У нас есть способы найти разные места внутри переменной

Что Не является «Коллекцией»?

Большинство переменных хранят внутри только одно значение. Когда мы присваиваем переменной новое значение, старое значение перезаписывается / заменяется

```
$ python
>>> x = 2
>>> x = 4
>>> print(x)
4
```

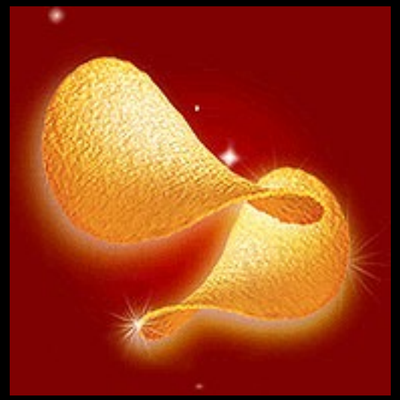
История двух коллекций..

- Список

  - Упорядоченный набор значений, сохраняющих свой порядок

- Словарь

  - «Сумка» со значениями, каждое из которых имеет собственную метку



Словари

https://ru.wikipedia.org/wiki/Ассоциативный_массив

Словари

- Словари — самый мощный инструмент в Пайтон для хранения данных
- Словари позволяют производить в Пайтон быстрые операции подобные работе с базой данных
- Словари имеют разные названия в других языках программирования:
  - Ассоциативный массив в Perl / PHP
  - Свойства, Мап или ХэшМап в Java
  - Контейнер свойств в C# / .Net

Словари

- Списки индексируют свои записи на основе позиции элемента в списке
- Словари же больше напоминают сумки — никакого порядка
- Поэтому мы обозначаем значения, которые кладем в словарь, с помощью «поисковой метки»

```
>>> purse = dict()
>>> purse['деньги'] = 12
>>> purse['конфеты'] = 3
>>> purse['платки'] = 75
>>> print(purse)
{'деньги': 12, 'платки': 75,
'конфеты': 3}
>>> print(purse['конфеты'])
3
>>> purse['конфеты'] =
purse['конфеты'] + 2
>>> print(purse)
{'деньги': 12, 'платки': 75,
'конфеты': 5}
```

Сравнение списков и словарей

Словари похожи на списки, за исключением того, что для поиска в них значений используются ключи, а не числа

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> ddd = dict()
>>> ddd['возраст'] = 21
>>> ddd['курс'] = 182
>>> print(ddd)
{'курс': 182, 'возраст': 21}
>>> ddd['возраст'] = 23
>>> print(ddd)
{'курс': 182, 'возраст': 23}
```

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> ddd = dict()
>>> ddd['возраст'] = 21
>>> ddd['курс'] = 182
>>> print(ddd)
{'курс': 182, 'возраст': 21}
>>> ddd['возраст'] = 23
>>> print(ddd)
{'курс': 182, 'возраст': 23}
```

Список

Ключ	Значение
[0]	21
[1]	183

lst

Словарь

Ключ	Значение
['курс']	182
['возраст']	21

ddd

Словарные литералы (Константы)

- Словарные литералы используют фигурные скобки и содержат список, состоящий из пар ключ : значение
- Создать пустой словарь можно с помощью пустых фигурных скобок

```
>>> jjj = { 'Чак' : 1 , 'Фред' : 42, 'Ян': 100}
>>> print(jjj)
{'Ян': 100, 'Чак': 1, 'Фред': 42}
>>> ooo = { }
>>> print(ooo)
{}
>>>
```

Какое имя встречается чаще?

Какое имя встречается чаще?

marquard

zhen

csev

zhen

cwen

marquard

zhen

csev

cwen

zhen

csev

marquard

zhen

Какое имя встречается чаще?

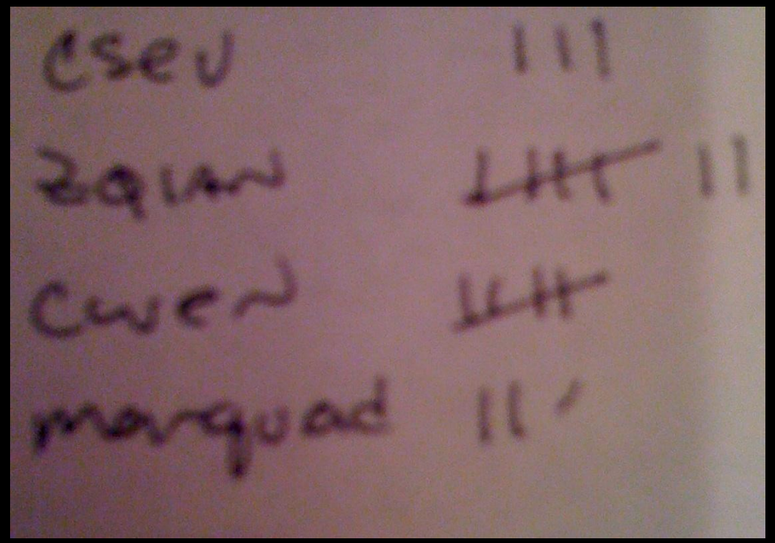
marquard

zhen

csev

zhen

cwen



csev

cwen

zhen

csev

marquard

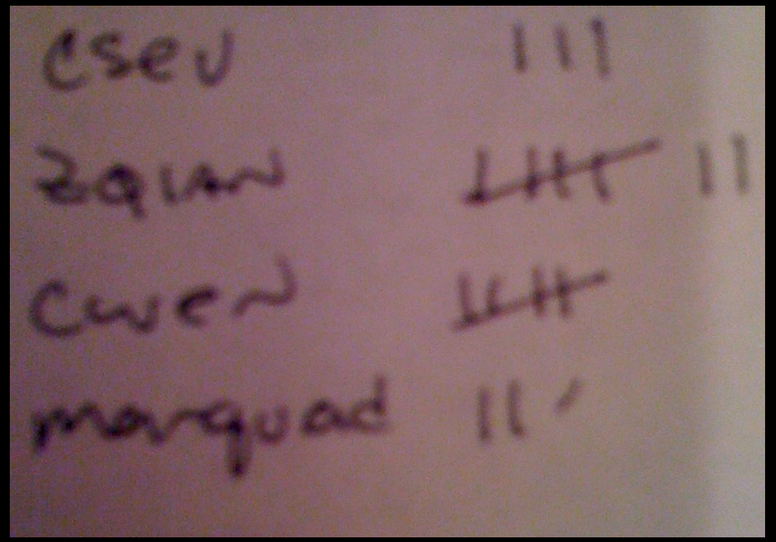
zhen

Словарь: подсчет количества

Словари часто используют для подсчета того, как часто встречается тот или иной элемент

```
>>> ccc = dict()
>>> ccc['csev'] = 1
>>> ccc['cwen'] = 1
>>> print(ccc)
{'csev': 1, 'cwen': 1}
>>> ccc['cwen'] = ccc['cwen'] + 1
>>> print(ccc)
{'csev': 1, 'cwen': 2}
```

Ключ Значение



Поиск по словарю

- Ошибкой будет ссылаться на ключ, которого нет в словаре
- Мы можем использовать оператор in, чтобы узнать, есть ли искомый ключ в словаре

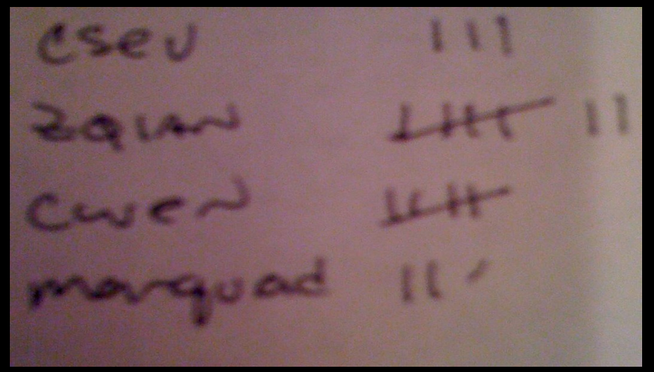
```
>>> ccc = dict()
>>> print(ccc['csev'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'csev'
>>> 'csev' in ccc
False
```

Когда нам встречается новое имя

Встретив новое имя, мы должны добавить новую запись в словарь.
А в случаях когда имя встретилось во второй и последующие разы, нужно просто добавить единицу к счётчику, относящемуся к этому имени в словаре

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    if name not in counts:
       counts[name] = 1
    else :
        counts[name] = counts[name] + 1
print(counts)
```

{'csev': 2, 'zqian': 1, 'cwen': 2}



Словари: метод get

Шаблон проверки наличия ключа в словаре, и установление значения по умолчанию, если ключ в словаре отсутствует, настолько распространён, что существует специальный метод get(), который делает это за нас

```
if name in counts:
    x = counts[name]
else :
    x = 0
```

```
x = counts.get(name, 0)
```

Оранжевым выделено значение по умолчанию в случае, если ключ отсутствует, а также нет трассировки (информации о произошедших ошибках)

{'csev': 2, 'zqian': 1, 'cwen': 2}

Упрощённый подсчёт с get()

Мы можем использовать метод get() и подставить значение по умолчанию равное нулю, если ключа еще нет в словаре. А затем добавлять единицу к этому значению

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    counts[name] = counts.get(name, 0) + 1
print(counts)
```

Значение
по умолчанию

{'csev': 2, 'zqian': 1, 'cwen': 2}

Упрощённый подсчёт с get()

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    counts[name] = counts.get(name, 0) + 1
print(counts)
```

http://www.youtube.com/watch?v=EHJ9uYx5L58

Подсчёт слов в тексте

Написание программ (программирование) — очень творческое и полезное занятие. Вы можете писать программы по разным причинам: желая заработать на жизнь, решить сложную задачу анализа данных или получить удовольствие от помощи кому-либо в решении проблемы. Эта книга предполагает, что каждому нужно знать, как программировать, а как только вы этому научитесь, обязательно найдете применение приобретенным навыкам.

В повседневной жизни нас окружают компьютеры, ноутбуки, мобильные телефоны. Мы можем воспринимать эти устройства как наших «персональных помощников», которые могут выполнять какие-то функции, облегчая нам жизнь. Аппаратное обеспечение современных компьютеров по сути устроено так, чтобы постоянно спрашивать: «Что бы вы хотели, чтобы я сделал дальше?»

Компьютеры быстры, обладают огромным объемом памяти и могли бы быть для нас очень полезны, знай мы язык, чтобы объяснить компьютеру, что мы хотели бы, чтобы он сделал дальше. Если бы мы знали этот язык, то могли бы переложить на компьютер выполнение некоторых повторяющихся задач. Забавно, что задачи, с которыми компьютеры справляются лучше всего, людям чаще всего кажутся скучными и утомительными.

Шаблон подсчета

```
counts = dict()
print('Введите строку текста:')
line = input('')

words = line.split()

print('Слова:', words)

print('Идет подсчет...')
for word in words:
    counts[word] =
counts.get(word,0) + 1
print('Итог:', counts)
```

Распространенный шаблон подсчета слов в строке текста состоит в том, чтобы с помощью метода split разделить строку на отдельные слова, затем перебрать их и, используя словарь, следить за появлением каждого слова независимо от других, и вести подсчет.

```
python wordcount.py
Введите строку текста:
the clown ran after the car and the car ran into the tent
and the tent fell down on the clown and the car

Слова: ['the', 'clown', 'ran', 'after', 'the', 'car',
'and', 'the', 'car', 'ran', 'into', 'the', 'tent', 'and',
'the', 'tent', 'fell', 'down', 'on', 'the', 'clown',
'and', 'the', 'car']
Идет подсчет…

Итог: {'and': 3, 'on': 1, 'ran': 2, 'car': 3, 'into': 1,
'after': 1, 'clown': 2, 'down': 1, 'fell': 1, 'the': 7,
'tent': 2}
```

http://www.flickr.com/photos/71502646@N00/2526007974/

```
counts = dict()
line = input('Введите строку текста:')
words = line.split()

print('Слова:', words)
print('Идет подсчет...’)

for word in words:
    counts[word] = counts.get(word,0) + 1
print('Итог:', counts)
```

```
python wordcount.py
Введите строку текста:
the clown ran after the car and the car ran
into the tent and the tent fell down on the
clown and the car

Слова: ['the', 'clown', 'ran', 'after', 'the', 'car',
'and', 'the', 'car', 'ran', 'into', 'the', 'tent', 'and',
'the', 'tent', 'fell', 'down', 'on', 'the', 'clown',
'and', 'the', 'car']
Идет подсчет...

Итог: {'and': 3, 'on': 1, 'ran': 2, 'car': 3, 'into':
1, 'after': 1, 'clown': 2, 'down': 1, 'fell': 1,
'the': 7, 'tent': 2}
```

Циклы со счетчиком и словари

Несмотря на то, что данные в словарях не упорядочены, мы можем написать цикл с оператором for, который просматривает все записи в словаре: на самом деле он просматривает все ключи в словаре и ищет значения

```
>>> counts = { 'Чак' : 1 , 'Фред' : 42, 'Ян': 100}
>>> for key in counts:
...     print(key, counts[key])
...
Ян 100
Чак 1
Фред 42
>>>
```

Получение списков ключей и значений

Вы можете получить из словаря список ключей (keys), значений (values) или элементов (ключ и значение) (items)

```
>>> jjj = { 'Чак' : 1 , 'Фред' : 42, 'Ян': 100}
>>> print(list(jjj))
['Ян', 'Чак', 'Фред']
>>> print(jjj.keys())
['Ян', 'Чак', 'Фред']
>>> print(jjj.values())
[100, 1, 42]
>>> print(jjj.items())
[('Ян', 100), ('Чак', 1), ('Фред', 42)]
>>>
```

Что такое «кортеж»? – об этом чуть позже...

Бонус: Две переменных цикла!

- Мы перебираем пары ключ-значение в словаре, используя *две* переменных цикла

- В каждой итерации первая переменная — это ключ, а вторая — соответствующее ключу значение

```
jjj = { 'Чак' : 1 , 'Фред' : 42, 'Ян': 100}
for aaa,bbb in jjj.items() :
    print(aaa, bbb)
```

```
Ян 100
Чак 1
Фред 42
```

aaa	bbb
[Ян]	100
[Чак]	1
[Фред]	42

```
name = input('Введите имя файла:')
handle = open(name)

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

```
python words.py
Enter file: words.txt
to 16
```

```
python words.py
Enter file: clown.txt
the 7
```

Использование двух вложенных циклов

Резюме

- Что такое коллекция?
- Списки vs. Словари
- Устройство словарей
- Самое частое слово
- Использование метода get()
- Хеширование и отсутствие порядка
- Написание циклов словаря
- Беглый взгляд на кортежи
- Сортировка словарей

Авторы / Благодарности

Исходная разработка: Чарльз Северанс, Школа Информации Мичиганского Университета.

Перевод выполнила Фомкина Виолетта.

… Добавьте сюда новых авторов и переводчиков

...