



РАНХиГС

РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА
И ГОСУДАРСТВЕННОЙ СЛУЖБЫ
ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

ЭМИТ



Чистый Python

Зам. Декана ЭФ ЭМИТ РАНХиГС,
проф. кафедры Эконометрики и математической экономики ЭМИТ РАНХиГС
д.т.н. Шилин Кирилл Юрьевич

РАНХиГС каб. 419/3

email: kshilin@ranepa.ru

Структурированные данные Python

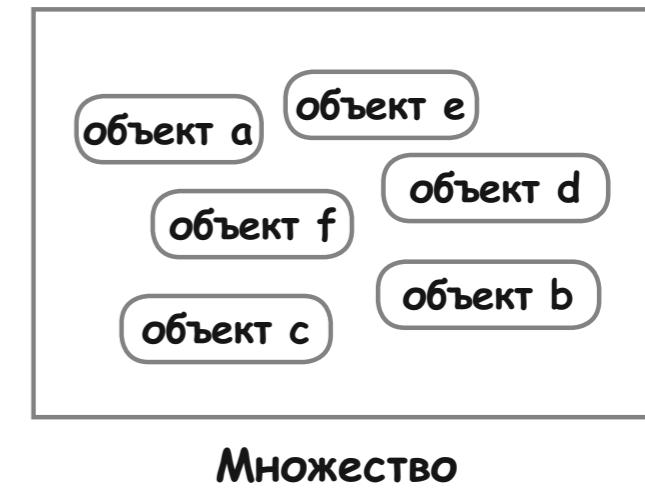
Списки могут динамически расти и сокращаться до любого размера.

Объекты в списке хранятся под индивидуальными номерами.



Как и в массивах, элементы списков нумеруются с нуля, и эти номера называются «значениями индекса» или просто «индексами».

Множество можно представить в виде коллекции неупорядоченных уникальных элементов – повторов нет.



Множество

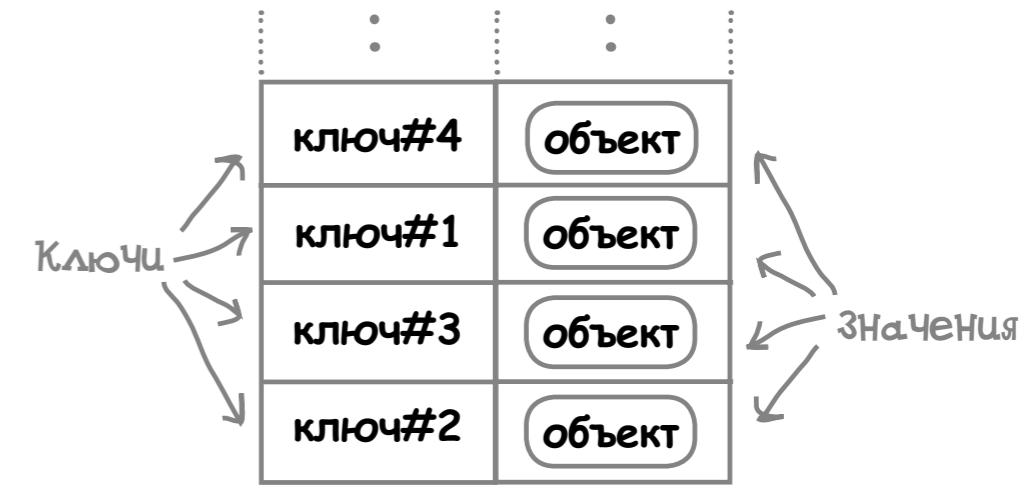
Кортежи похожи на списки, только их нельзя изменить.

Кортежи – константные списки.



В кортежах также есть индексы (как и в списках).

Словари связывают ключи со значениями, и (как списки) их можно динамически сокращать или увеличивать до любого размера.



Словарь



Список в Python

```
prices = []
```

Имя переменной находится слева от оператора присваивания...

...а лiteralный список – справа. В этом примере список пуст.

```
temps = [ 32.0, 212.0, 0.0, 81.6, 100.0, 45.3 ]
```

Объекты (в данном случае числа с плавающей точкой) отделены друг от друга запятыми и заключены в квадратные скобки – это список.

```
words = [ 'hello', 'world' ]
```

Список строковых объектов.

```
car_details = [ 'Toyota', 'RAV4', 2.2, 60807 ]
```

Список объектов различных типов.

```
everything = [ prices, temps, words, car_details ]
```

Списки внутри списка.

```
odds_and_ends = [ [ 1, 2, 3], ['a', 'b', 'c' ],  
[ 'One', 'Two', 'Three' ] ]
```

Списки НЕКОПИЯ

```
first → [1, 2, 3, 4, 5] ← second
```

`>>> first = [1, 2, 3, 4, 5]` Создать новый список (и положить в него пять числовых объектов).

`>>> first` Пять элементов в списке «first».

`[1, 2, 3, 4, 5]`

`>>> second = first` «Скопировать» существующий список в новый, который назовем «second».

`>>> second` Пять элементов в списке «second».

`[1, 2, 3, 4, 5]`

`>>> second.append(6)` Кажется, что все нормально. Но это не так.

`>>> second` Упс! Новый объект добавился также в список «first»!

`[1, 2, 3, 4, 5, 6]`



Списки копия



```
>>> third = second.copy()
>>> third
[1, 2, 3, 4, 5, 6]
```



Список «third»
вырос на один
объект.

```
>>> third.append(7)
>>> third
[1, 2, 3, 4, 5, 6, 7]
>>> second
[1, 2, 3, 4, 5, 6]
```

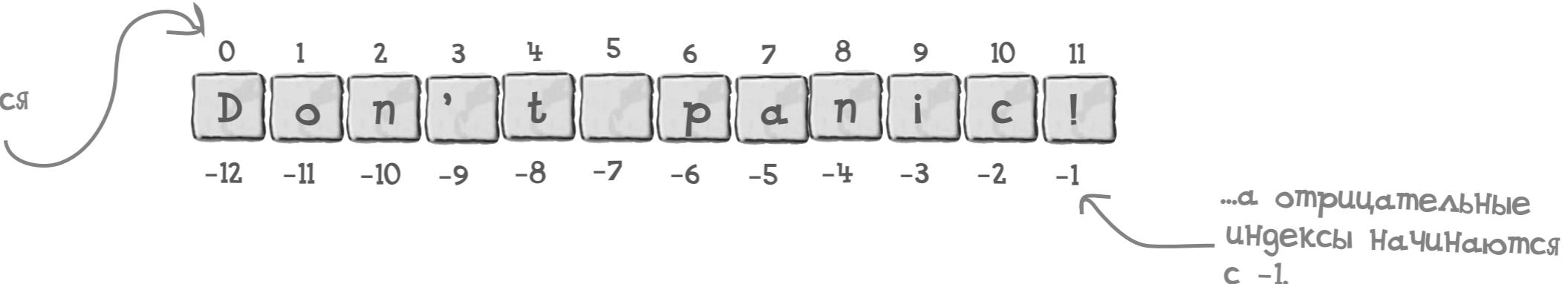


Намного лучше.
Существующий список
не изменился.

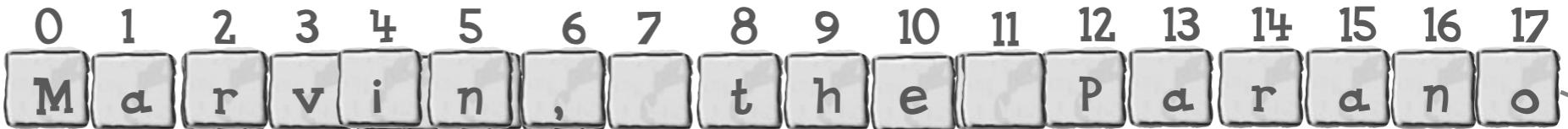
А вот это
похоже на правду.
Новый объект
добавлен только
в список «third»,
но не в остальные два
(``first`` и ``second``).

Адреса в списках

В списках Python
положительные
индексы начинаются
с 0...



letters →



Вспомним, что
доступ к любому
объекту в списке
можно получить
по положительному
или отрицательному
индексу. Здесь
рассмотрим некоторые
отрицательные индексы.

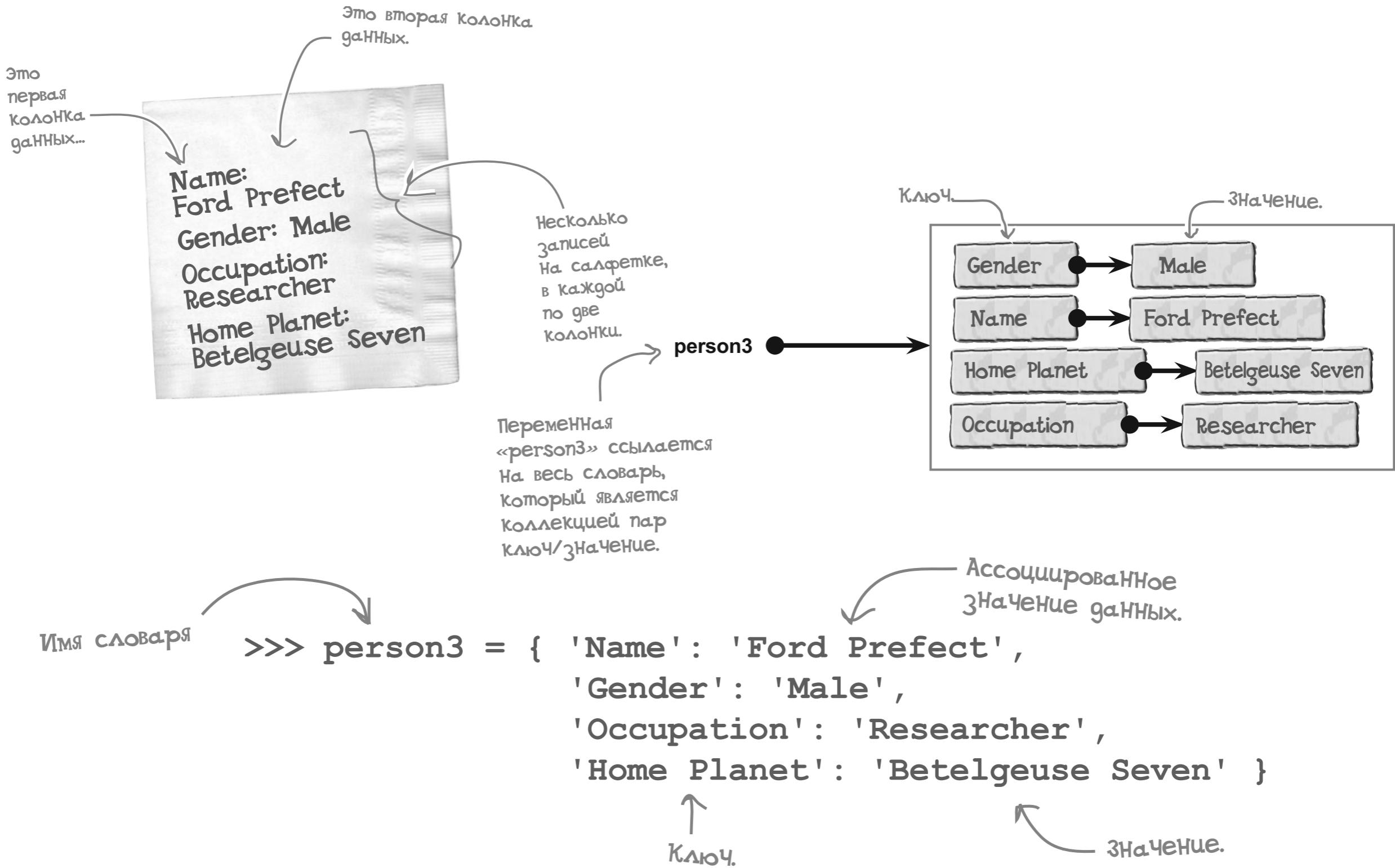
```
for char in letters[:6]:
    print('\t', char)
for char in letters[-7:]:
    print('\t'*2, char)
for char in letters[12:20]:
    print('\t'*3, char)
```

letters[:6] → Marvin

letters[-7:] → Android

letters[12:20] → Paranoid

Словари в Python



Словари: отсутствие порядка

•	•	•	•
ключ №4	объект		
ключ №1	объект		
ключ №3	объект		
ключ №2	объект		

Словарь

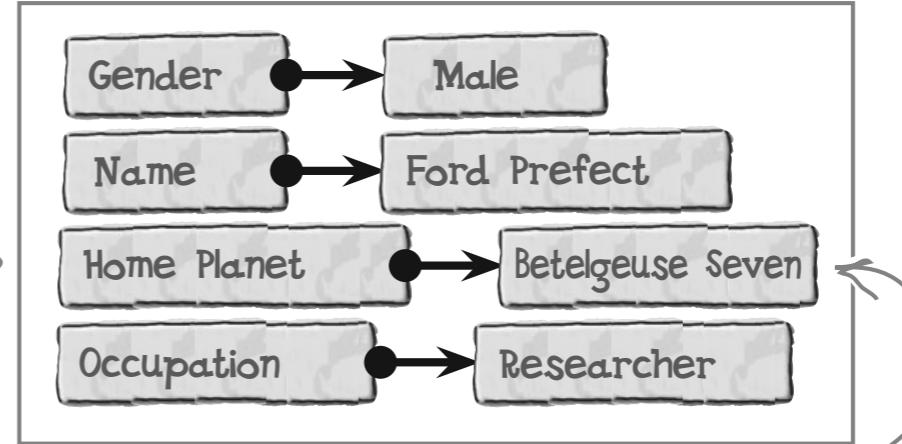
```
>>> person3 = { 'Name': 'Ford Prefect',  
                 'Gender': 'Male',  
                 'Occupation': 'Researcher',  
                 'Home Planet': 'Betelgeuse Seven' }  
  
>>> person3  
{'Gender': 'Male', 'Name': 'Ford Prefect', 'Home Planet':  
     'Betelgeuse Seven', 'Occupation': 'Researcher'}
```

Мы ввели данные
в словарь в этом
порядке...

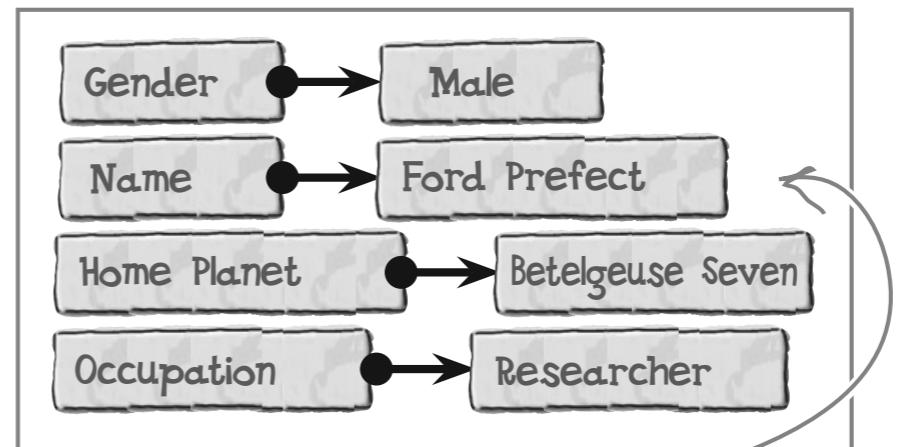
...а интерпретатор
использует другой
порядок.

Словари адресация

>>> person3['Home Planet']
'Betelgeuse Seven'



>>> person3['Name']
'Ford Prefect'



Укажите ключ
между квадратными
скобками.

>>> person3['Home Planet']
'Betelgeuse Seven'

значение данных,
ассоциированное
с ключом.

>>> person3['Name']
'Ford Prefect'

Множества

Обратите внимание
на порядок. Он
отличается от порядка
ввода. Также удалены
повторения.

```
>>> vowels = { 'a', 'e', 'e', 'i', 'o', 'u', 'u' }  
>>> vowels  
{'e', 'u', 'a', 'i', 'o'}
```

Множество начинается и заканчивается фигурными скобками.

Объекты отделяются друг от друга запятыми.

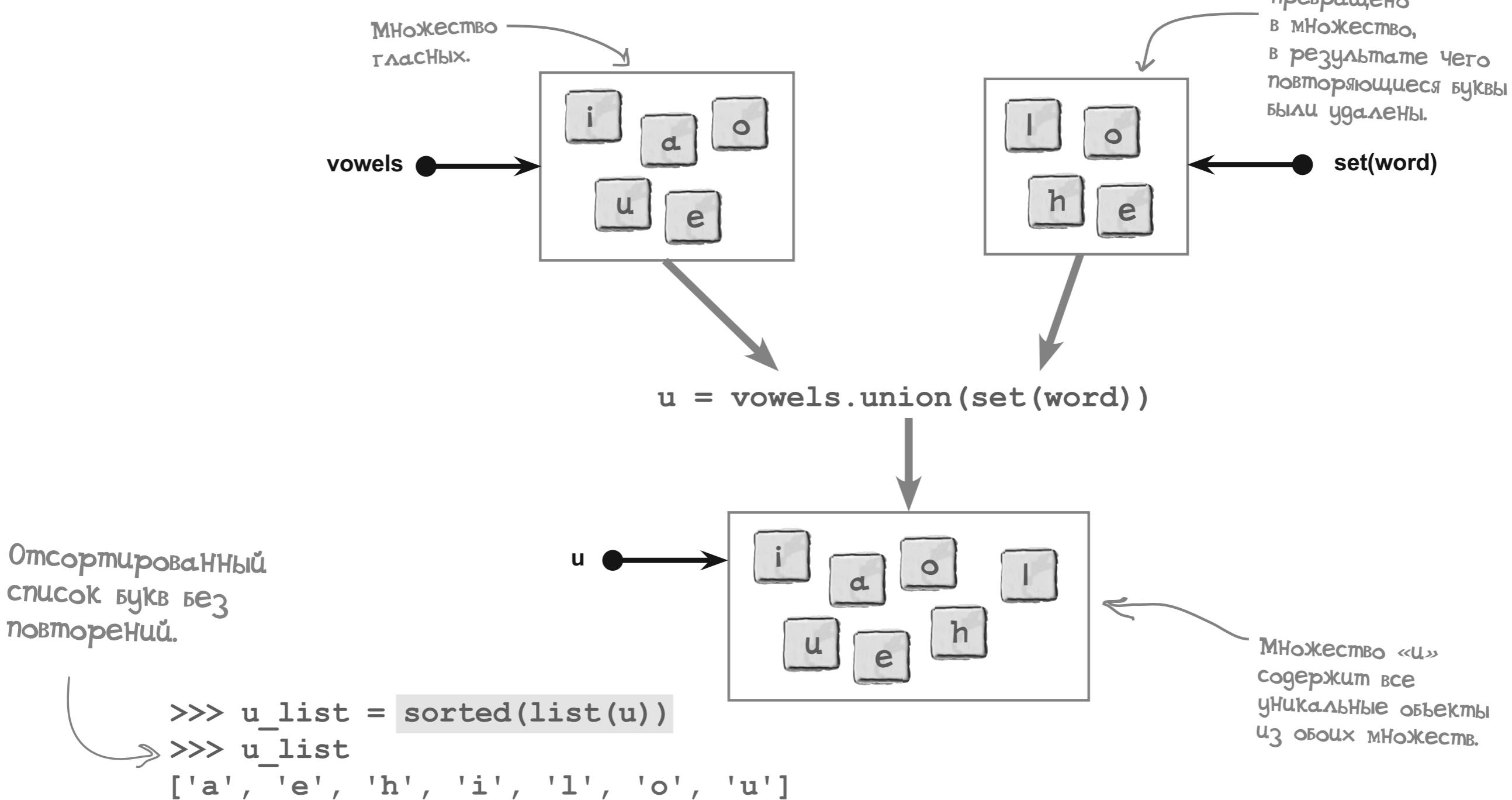
```
>>> vowels = { 'a', 'e', 'e', 'i', 'o', 'u', 'u' }  
>>> vowels  
{'e', 'u', 'a', 'i', 'o'}
```

Эти две строки кода делают одно и то же: они обе присваивают переменной новый объект множества.

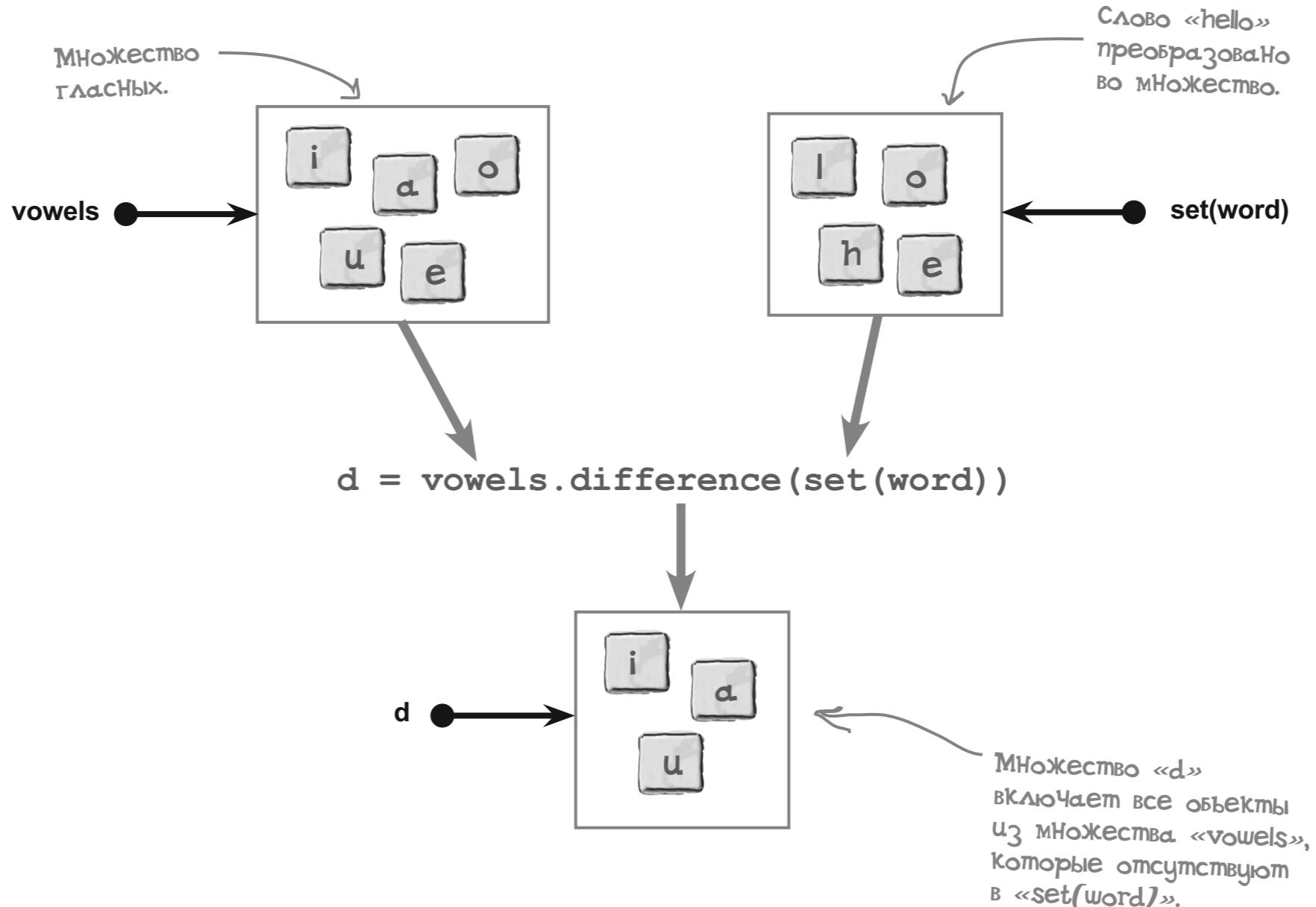
```
>>> vowels2 = set('aeiouuu')  
>>> vowels2  
{'e', 'u', 'a', 'i', 'o'}
```

Объединение множеств

```
>>> vowels = set('aeiou')
>>> word = 'hello'
```

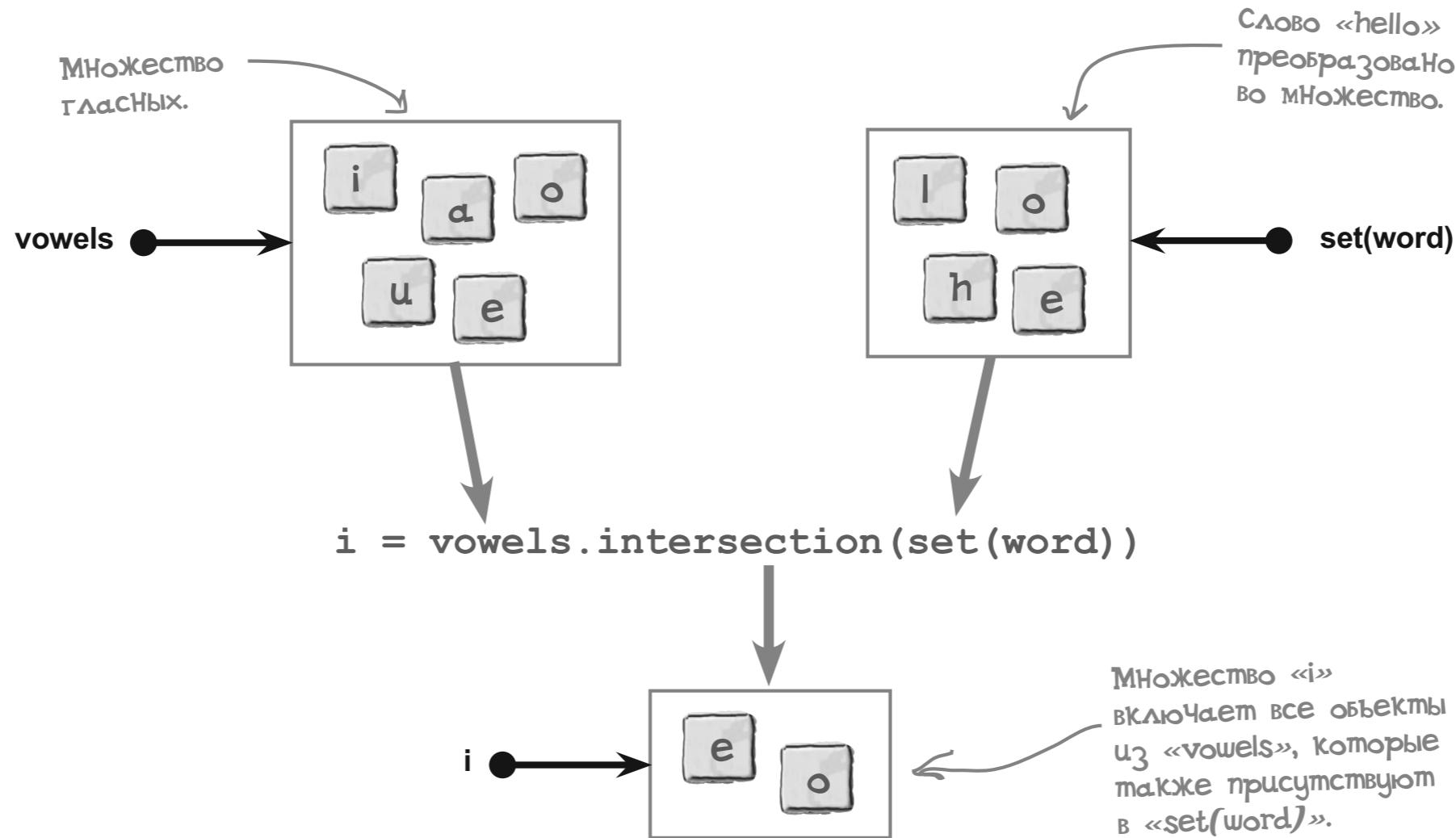


Различие в множествах



```
>>> d = vowels.difference(set(word))
>>> d
{'u', 'i', 'a'}
```

Общие объекты множеств



```
>>> i = vowels.intersection(set(word))
>>> i
{'e', 'o'}
```

Кортежи

Ничего нового.
Создан список
гласных.

```
>>> vowels = [ 'а', 'е', 'и', 'о', 'у' ]
>>> type(vowels)
<class 'list'>
>>> vowels2 = ( 'а', 'е', 'и', 'о', 'у' )
>>> type(vowels2)
<class 'tuple'>
```

Встроенная
функция «type»
сообщает тип
объекта.

Кортеж похож на список,
но не является им. Кортежи
заключены в круглые
(а не в квадратные) скобки.

```
>>> vowels
['а', 'е', 'и', 'о', 'у']
>>> vowels2
('а', 'е', 'и', 'о', 'у')
```

Круглые скобки показывают,
что это кортеж.

```
>>> vowels[2] = 'И'
>>> vowels
['а', 'е', 'И', 'о', 'у']
```

Присвоим букву «И»
верхнего регистра
третьему элементу
списка «vowels».

```
>>> vowels2[2] = 'И'
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    vowels2[2] = 'И'
```

Интерпретатор
громко возмущается
в ответ на попытку
изменить кортеж.

```
TypeError: 'tuple' object does not support item assignment
>>> vowels2
('а', 'е', 'и', 'о', 'у')
```

Ничего не изменилось, потому
что кортежи неизменяемы.

Повторим: список и словарь

```
>>> l = list() ← Используем встроенную функцию
Пустой           «list» для создания пустого
список.          списка, а затем присвоим данные.  

>>> l  

[]  

>>> l = [ 1, 2, 3 ] ←  

>>> l  

[1, 2, 3]
```



```
>>> d = dict() ← Используем встроенную функцию
Пустой           «dict» для определения пустого
словаря.          словаря, а затем присвоим данные.  

{}  

>>> d = { 'first': 1, 'second': 2, 'third': 3 }  

>>> d  

{'second': 2, 'third': 3, 'first': 1}
```

Повторим: множество и кортеж

Пустое множество.

```
>>> s = set()  
>>> s  
set()  
>>> s = {1, 2, 3}  
>>> s  
{1, 2, 3}
```

Используем встроенную функцию «`set()`» для определения пустого множества, а затем присвоим данные.

Пустой кортеж

```
>>> t = tuple()  
>>> t  
()  
>>> t = (1, 2, 3)  
>>> t  
(1, 2, 3)
```

Используем встроенную функцию «`tuple()`» для создания пустого кортежа, а затем присвоим данные.

Множества заключаются в фигурные скобки, так же как словари. Однако пара фигурных скобок уже используется для представления пустого словаря, поэтому пустое множество представлено как «`set()`».