



# РАНХиГС

РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА  
И ГОСУДАРСТВЕННОЙ СЛУЖБЫ  
ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ



РАНХиГС  
экономический  
факультет



# Anaconda и Python

проф. кафедры Эконометрики и математической экономики ЭМИТ РАНХиГС  
д.т.н. Шилин Кирилл Юрьевич

РАНХиГС каб. 419/3  
email: [kshilin@ranepa.ru](mailto:kshilin@ranepa.ru)

# Дистрибутив Anaconda



Загрузить дистрибутив Anaconda: <https://www.anaconda.com/products/individual>



Версия Python 3.8

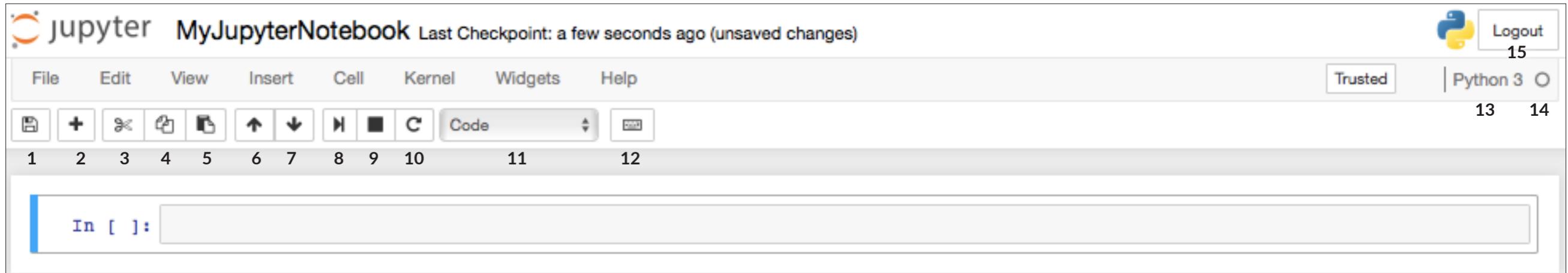


Редактор Jupyter Notebooks версия 6 и старше

# Jupyter Notebooks

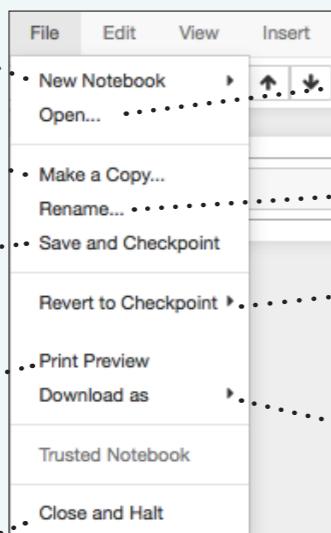
## Шпаргалка (Cheat Sheet) для Jupyter Notebook [ссылка](#)

Command Mode:



### Saving/Loading Notebooks

Create new notebook



Make a copy of the current notebook

Open an existing notebook

Rename notebook

Revert notebook to a previous checkpoint

Download notebook as  
- IPython notebook  
- Python  
- HTML  
- Markdown  
- reST  
- LaTeX  
- PDF

Save current notebook and record checkpoint

Preview of the printed notebook

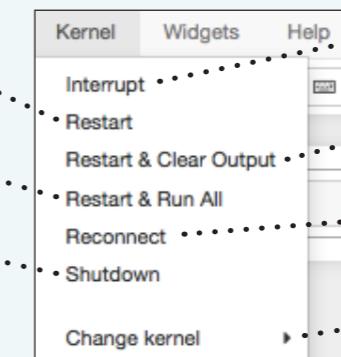
Close notebook & stop running any scripts

### Insert Cells

Add new cell above the current one



Add new cell below the current one



Interrupt kernel

Interrupt kernel & clear all output

Connect back to a remote notebook

Run other installed kernels



# Jupyter Notebooks

## Основные возможности:

1. Поля для ввода кода
2. Поблочное выполнение кода
3. Поля для ввода заголовков
4. Поля для ввода текста с формулами LaTeX

**Внимание!** команда для выполнения блока **<Shift>+<Enter>**

Ввод данных в режиме **Markdown**

## Заголовки

Заголовок 1 уровня: # или <h1>...</h1>

Заголовок 2 уровня: ## или <h2>...</h2>

Заголовок 3 уровня: ### или <h3>...</h3>

**Формула TeX:** в выделенной строке  $\int_0^\pi \sin^2(x) dx$   
внутри строки  $\int_0^\pi \sin^2(x) dx$

**Конец абзаца:** В конце абзаца поставить <br>



**РАНХиГС**

РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА  
И ГОСУДАРСТВЕННОЙ СЛУЖБЫ  
ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ



РАНХиГС  
экономический  
факультет



# Структурированные данные



# Структурированные данные Python

Списки могут динамически расти и сокращаться до любого размера.

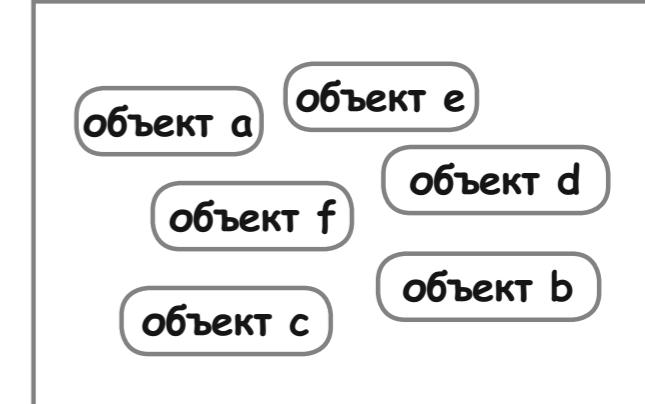
Объекты в списке хранятся под индивидуальными номерами.



Список

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.



кортеж

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' ] ]
```



```
>>> first = [1, 2, 3, 4, 5] ← Создать новый список  
(и положить в него пять  
числовых объектов).  
>>> first  
[1, 2, 3, 4, 5] ← Пять элементов в списке «first».  
>>> second = first ← «Скопировать» существующий список  
в новый, который назовем «second».  
>>> second  
[1, 2, 3, 4, 5] ← Пять элементов в списке «second».  
  
>>> second.append(6)  
>>> second  
[1, 2, 3, 4, 5, 6] ← Кажется, что все  
нормально. Но это не так.  
  
>>> first  
[1, 2, 3, 4, 5, 6] ← Упс! Новый объект  
добавился также  
в список «first»!
```



# 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]
```



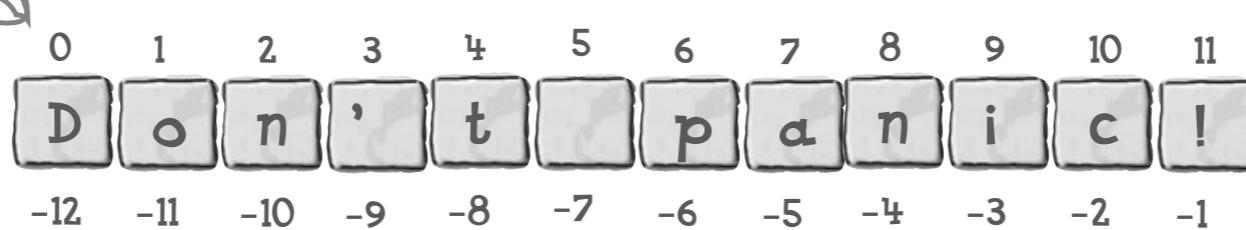
Намного лучше.  
Существующий список  
не изменился.

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...



...а отрицательные  
индексы начинаются  
с -1.



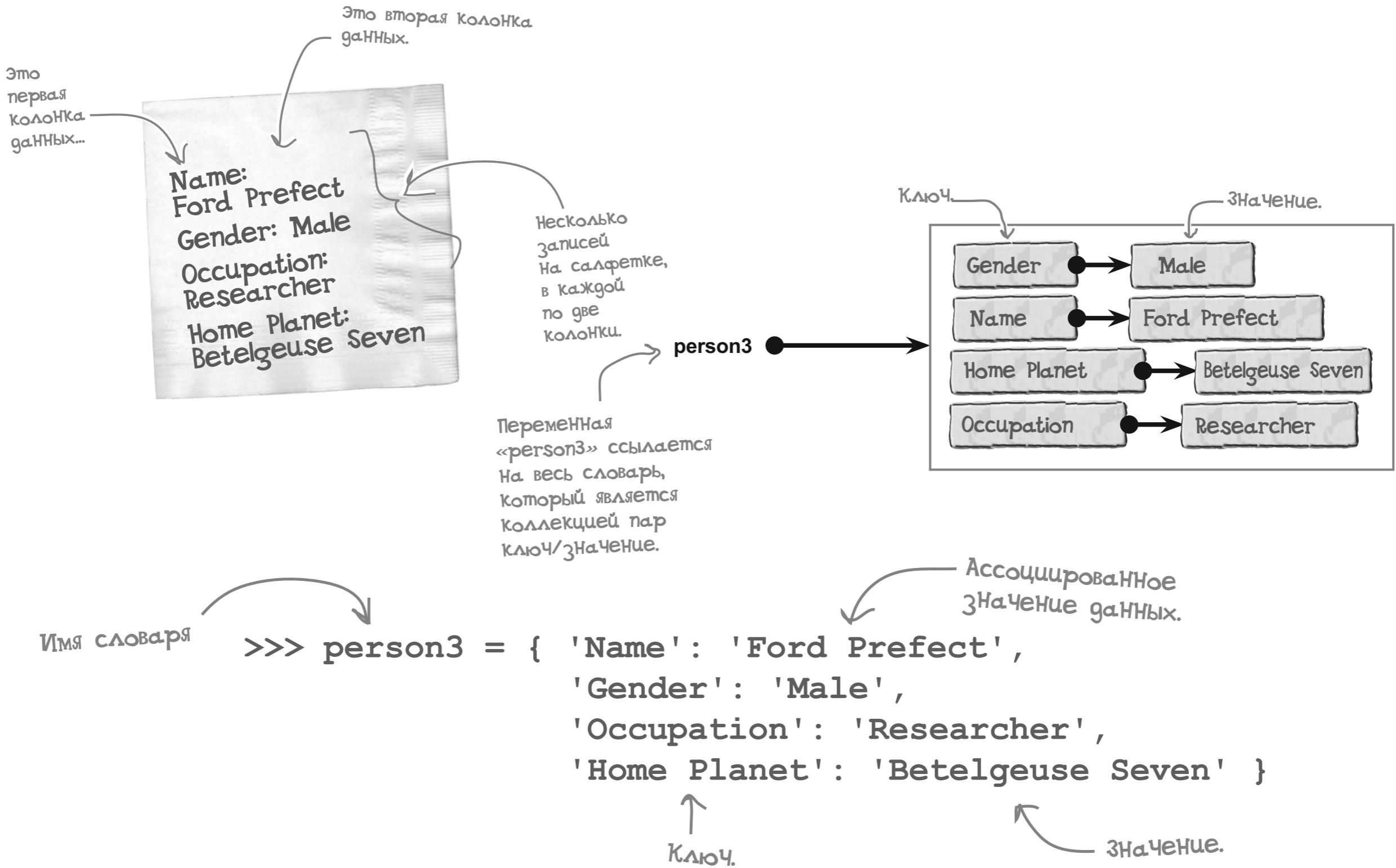
Вспомним, что  
доступ к любому  
объекту в списке  
можно получить  
по положительному  
или отрицательному  
индексу. Здесь  
рассмотрим некоторые  
отрицательные индексы.

```
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)
```





# 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'}
```

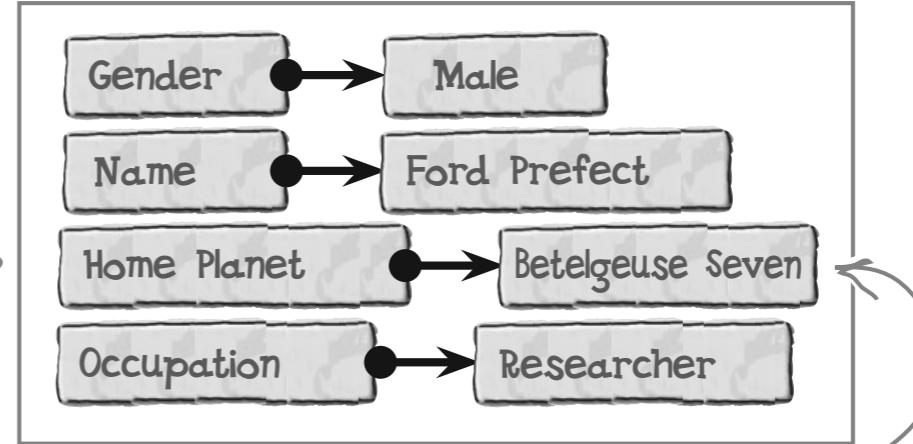
Мы ввели данные  
в Словарь в этом  
порядке...

...а интерпретатор  
использует другой  
порядок.

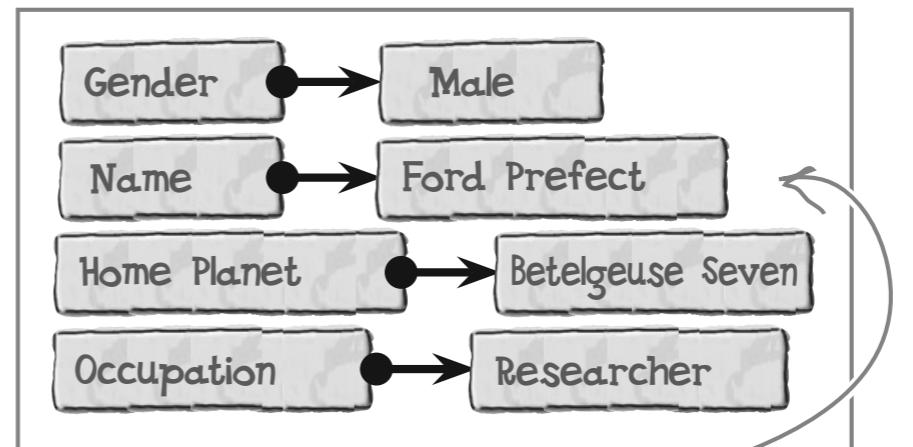


# Словари адресация

>>> 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.

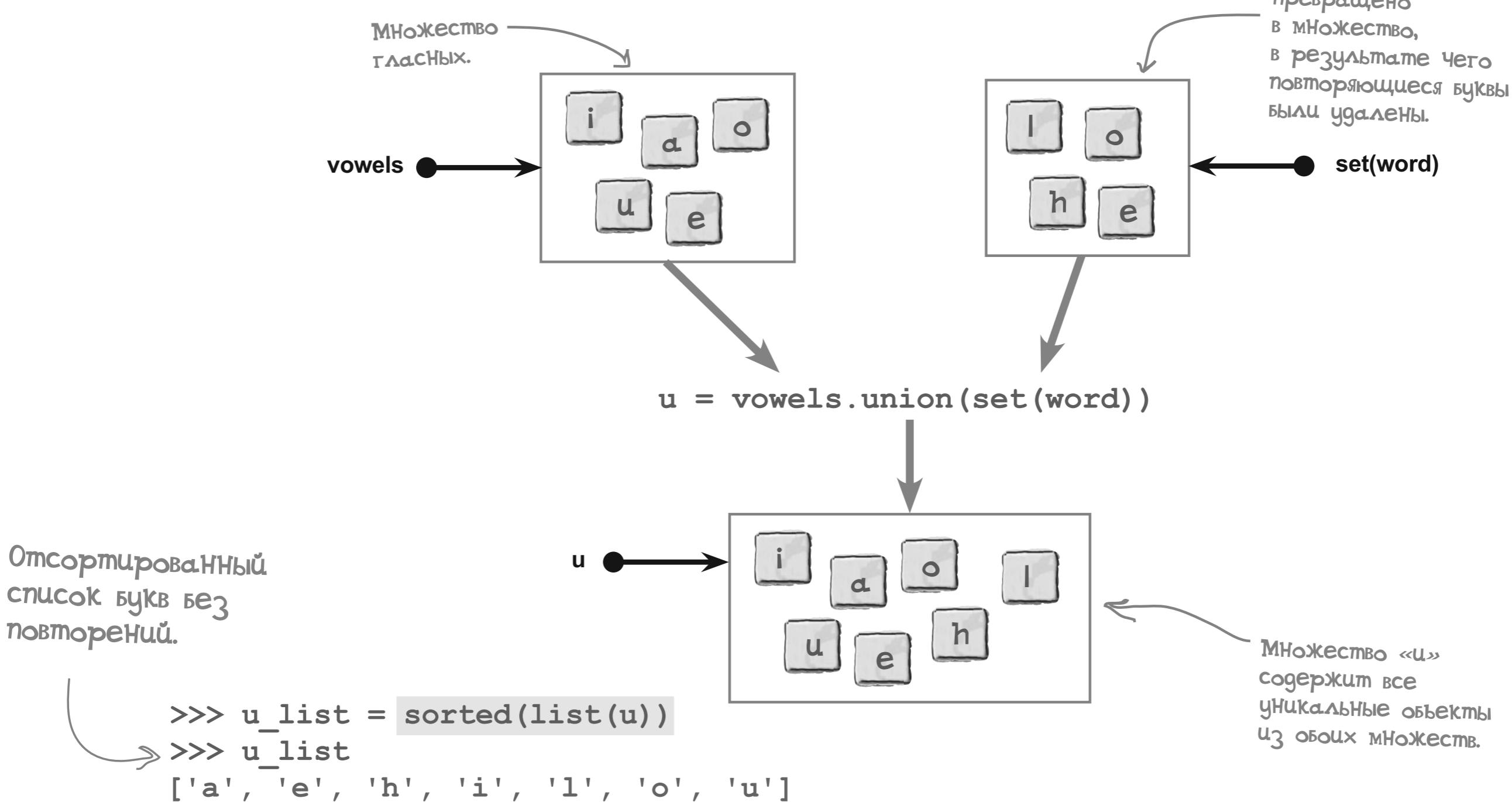
```
>>> vowels = { 'a', 'e', 'e', 'i', 'o', 'u', 'u' }  
>>> vowels  
{'e', 'u', 'a', 'i', 'o'}
```

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

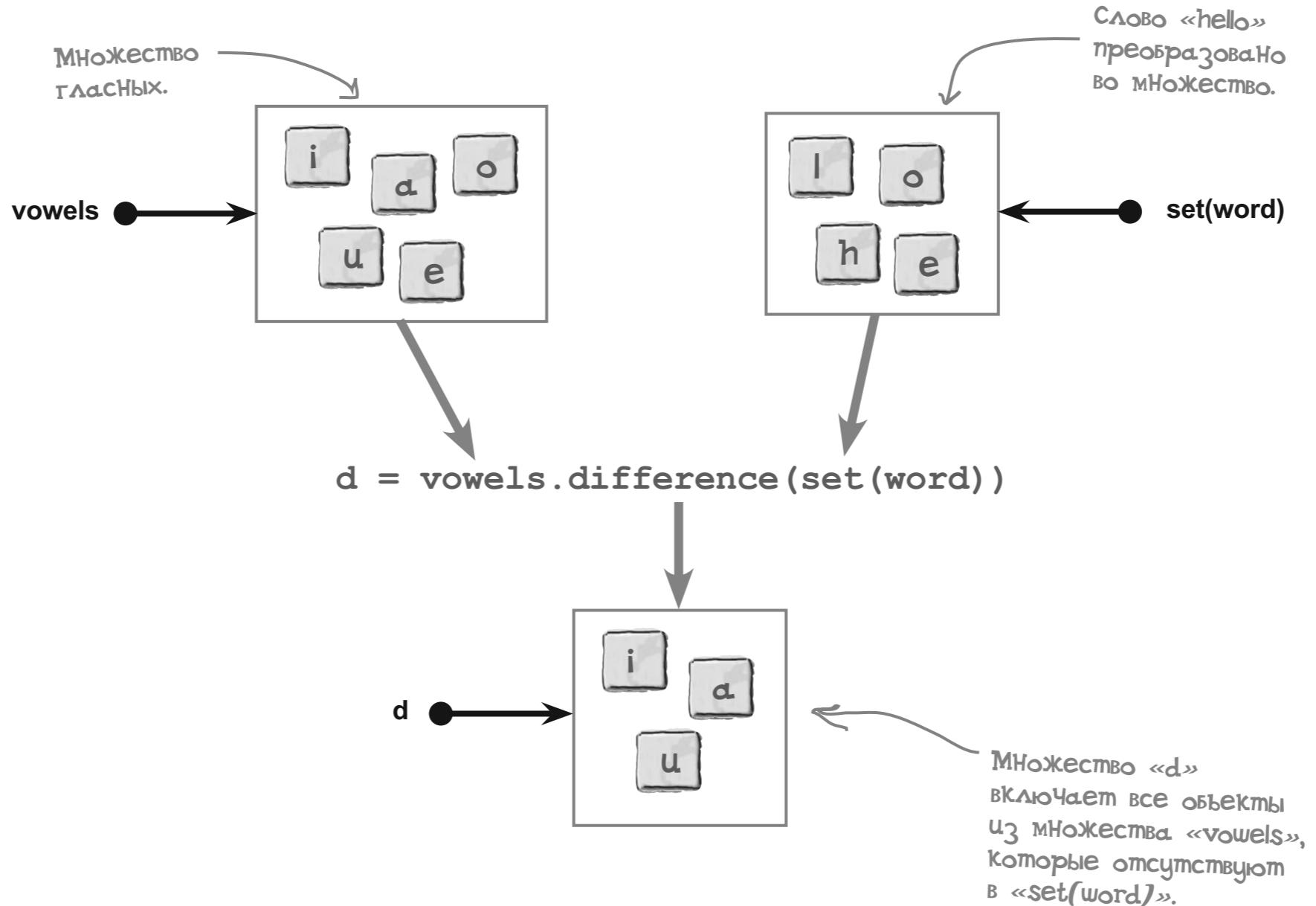
```
>>> vowels2 = set('aeeeiouuu')  
>>> vowels2  
{'e', 'u', 'a', 'i', 'o'}
```

# Объединение множеств

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



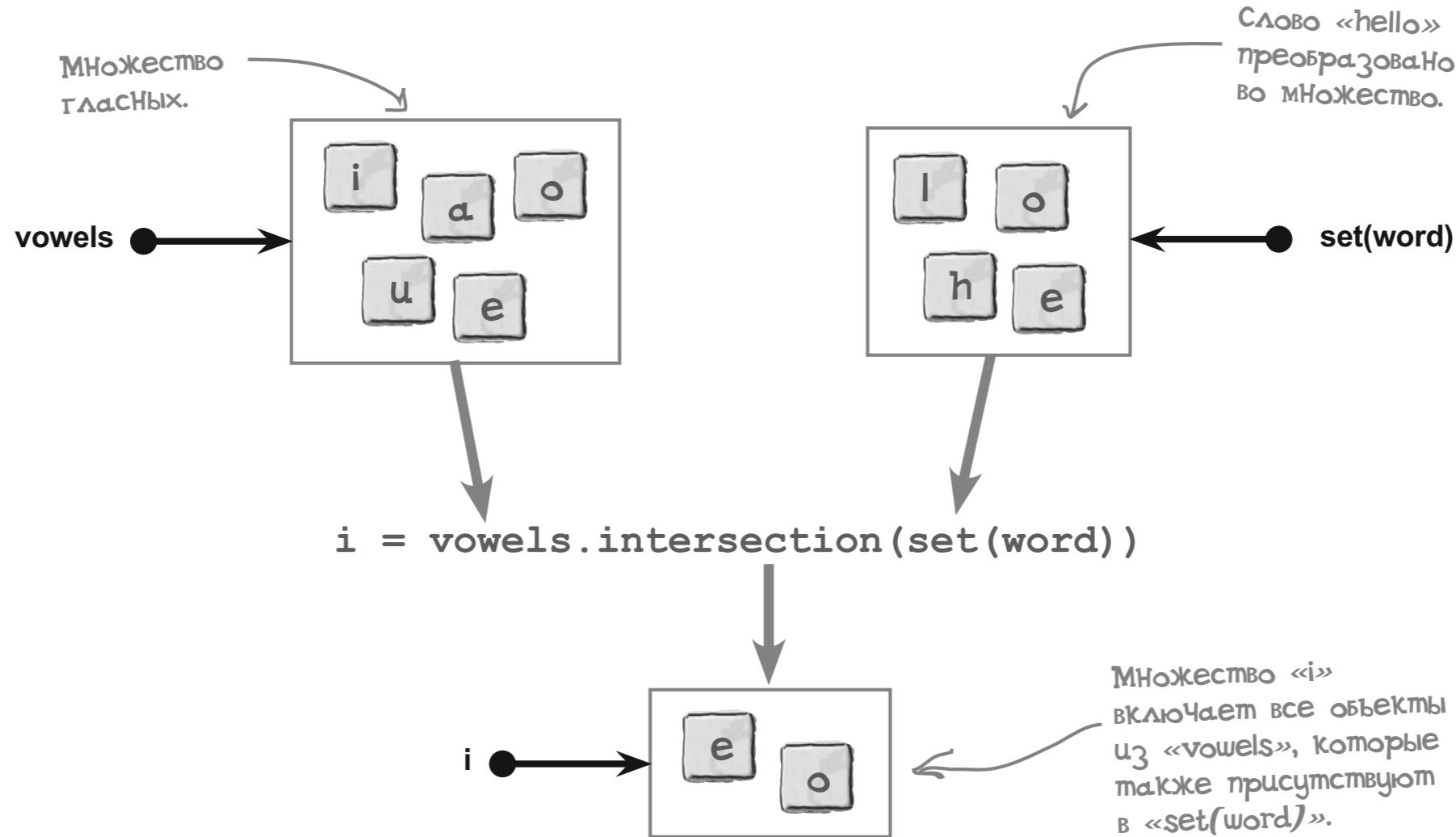
# Различие в множествах



```
>>> 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.

Встроенная  
функция «type»  
сообщает тип  
объекта.

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

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

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

Круглые скобки показывают,  
что это кортеж.

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

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

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

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

Пустой список.

```
>>> l = list()  
>>> l  
[]  
>>> l = [ 1, 2, 3 ]  
>>> l  
[1, 2, 3]
```

Используем встроенную функцию «list» для создания пустого списка, а затем присвоим данные.

Пустой словарь.

```
>>> d = dict()  
>>> d  
{ }  
>>> d = { 'first': 1, 'second': 2, 'third': 3 }  
>>> d  
{ 'second': 2, 'third': 3, 'first': 1}
```

Используем встроенную функцию «dict» для определения пустого словаря, а затем присвоим данные.

# 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()`».