



ПРЕЗИДЕНТСКАЯ
АКАДЕМИЯ

ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ

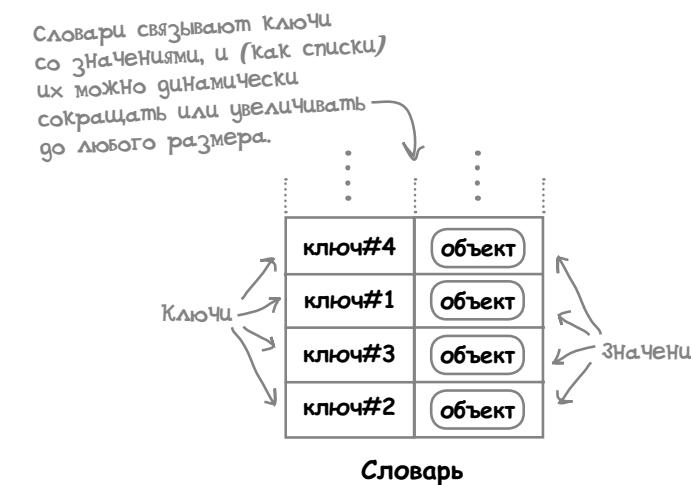
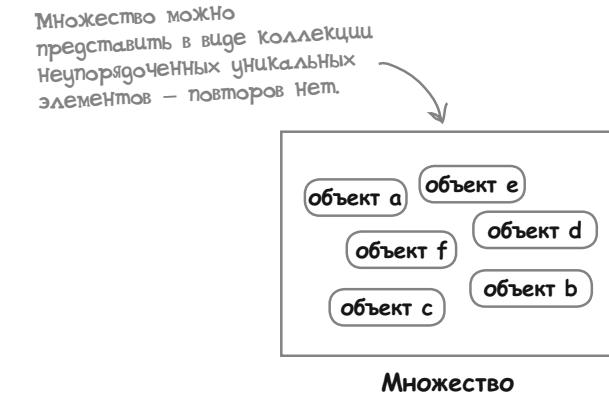
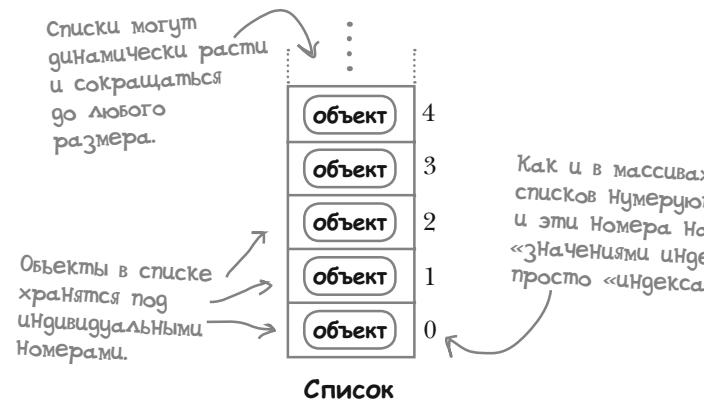
Структурированные данные

Москва, 2023

РАНХиГС

2023

Структурированные данные Python



Списки

```
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.

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

Список объектов различных типов.

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

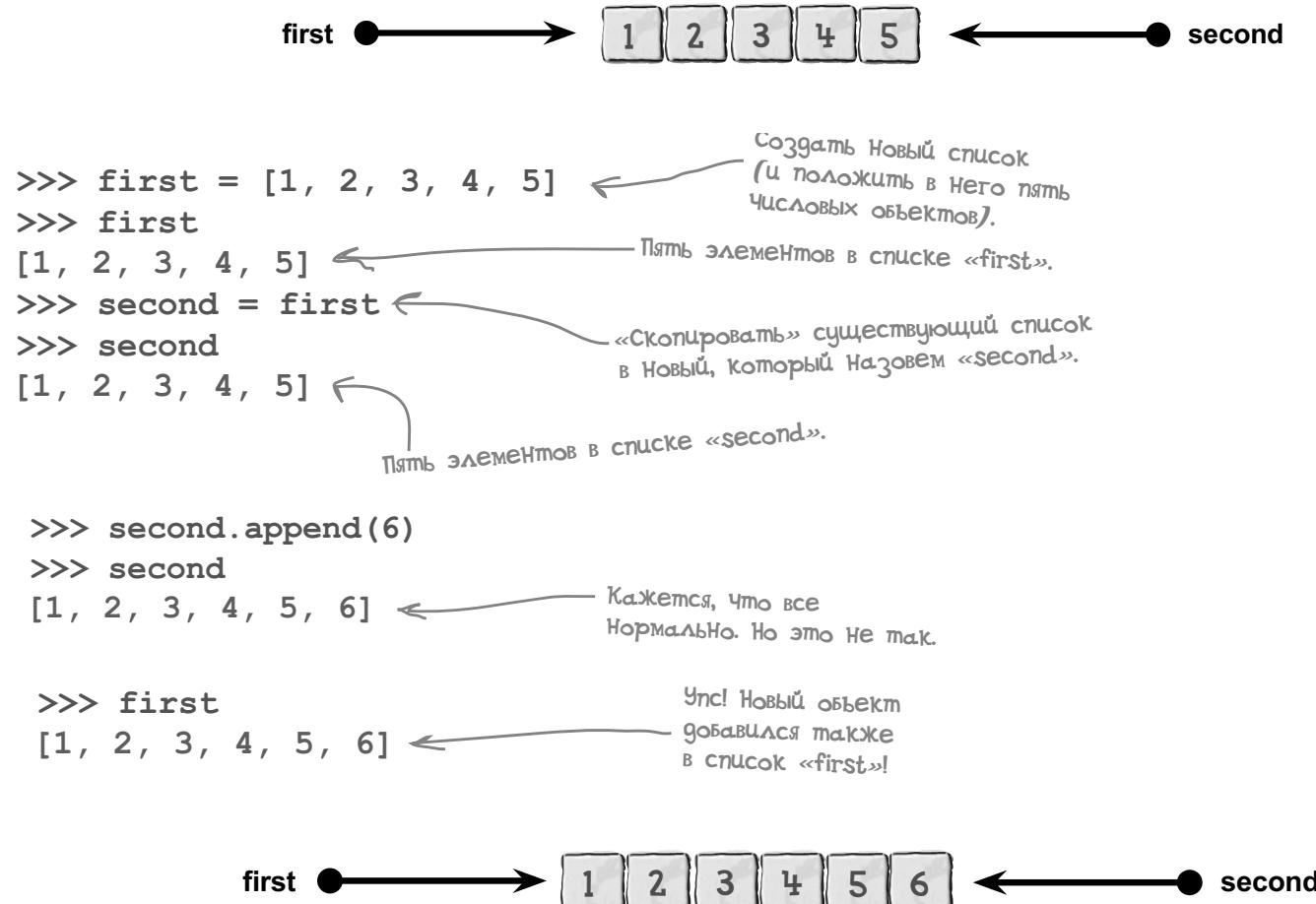
Список строковых объектов.

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

Списки внутри списка.

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

Что происходит при добавлении нового элемента?



Разрываем связь в списках



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

Список «third»
вырос на один
объект.

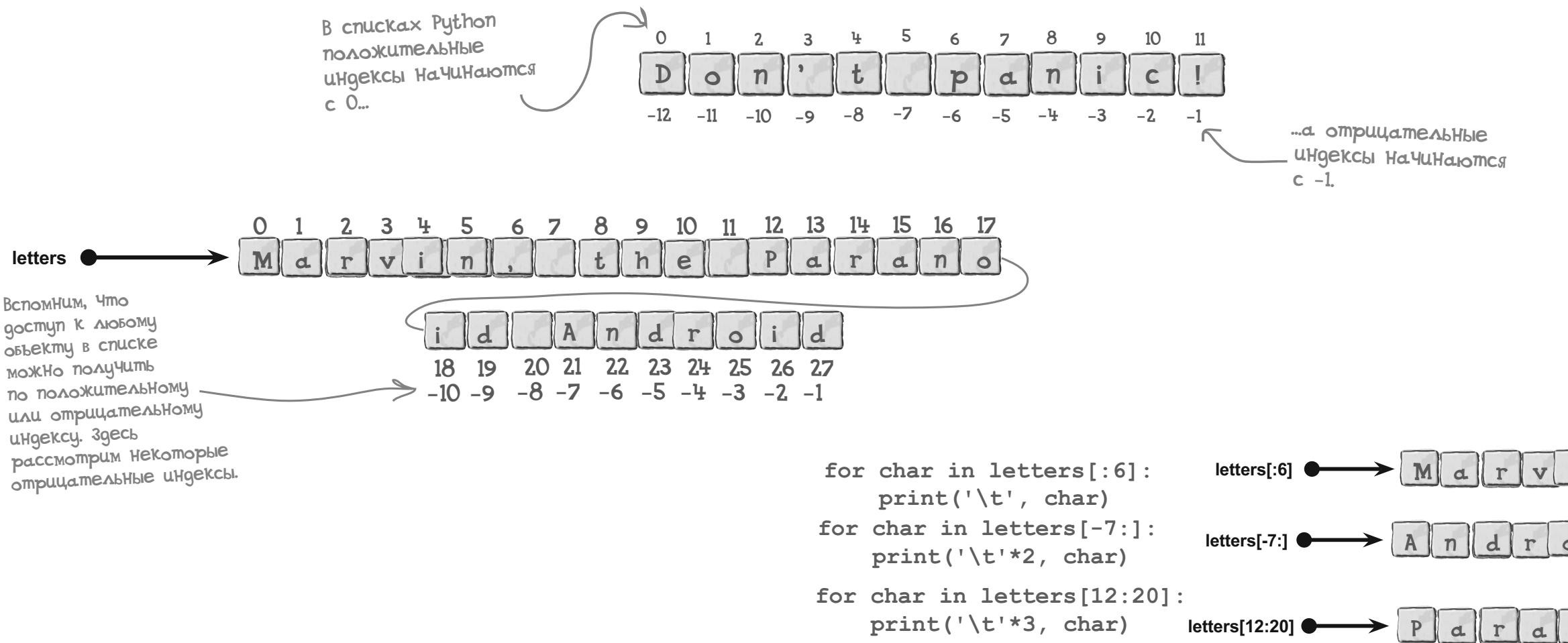
```
>>> third.append(7)
>>> third
[1, 2, 3, 4, 5, 6, 7]
>>> second
[1, 2, 3, 4, 5, 6]
```



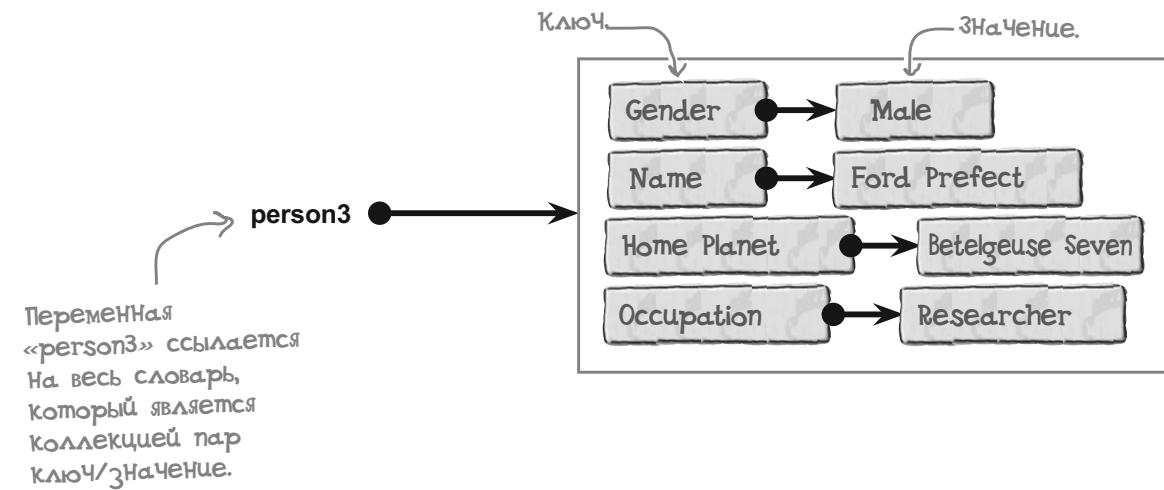
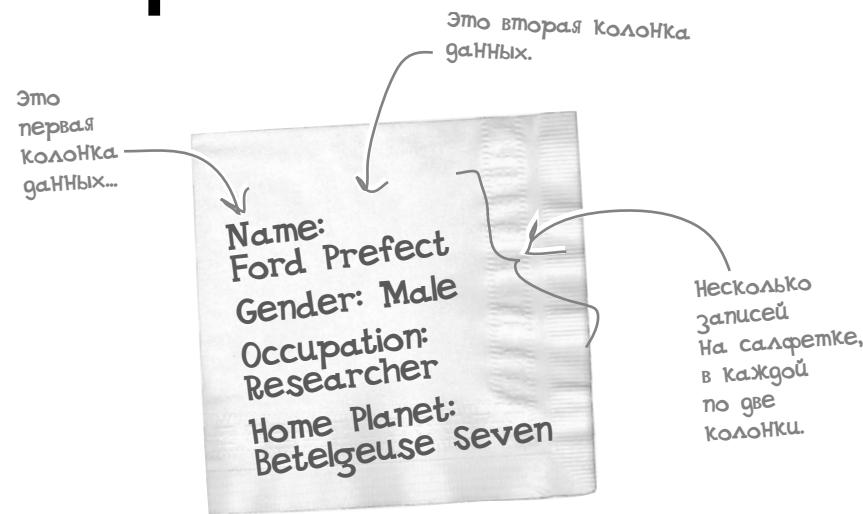
Намного лучше.
Существующий список
не изменился.

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

Разностная адресация



Словари



Имя словаря

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

↑
Ключ.

Ассоциированное значение данных.

↑
Значение.

Отсутствие порядка в словарях

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

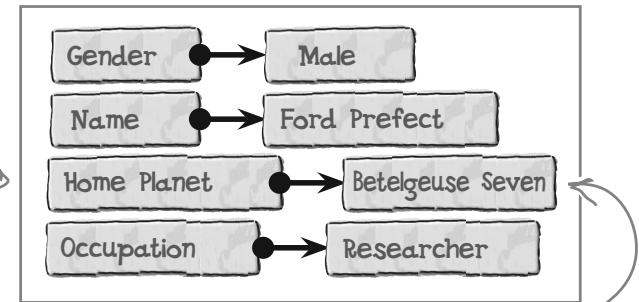
```
>>> person3 = { 'Name': 'Ford Prefect',  
                 'Gender': 'Male',  
                 'Occupation': 'Researcher',  
                 'Home Planet': 'Betelgeuse Seven' }  
  
>>> person3  
{'Gender': 'Male', 'Name': 'Ford Prefect', 'Home Pl  
'Betelgeuse Seven', 'Occupation': 'Researcher'}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

Мы ввели данные
в словарь в этом
порядке...

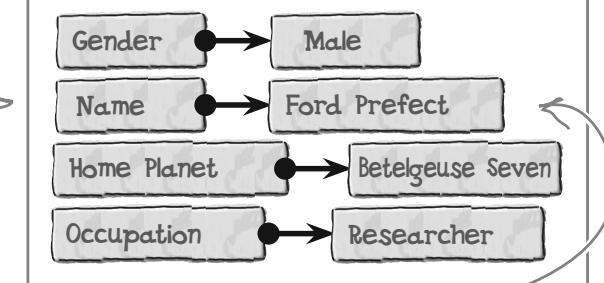
...а интерпретатор использует другой порядок.

Адресация в словаре

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



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



Укажите ключ
между квадратными
скобками.

```
>>> person3['Home Planet']  
'Betelgeuse Seven'  
>>> person3['Name']  
'Ford Prefect'
```

Значение данных,
ассоциированное
с ключом.

Множества

Обратите внимание
на порядок. Он
отличается от порядка
ввода. Так же 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.

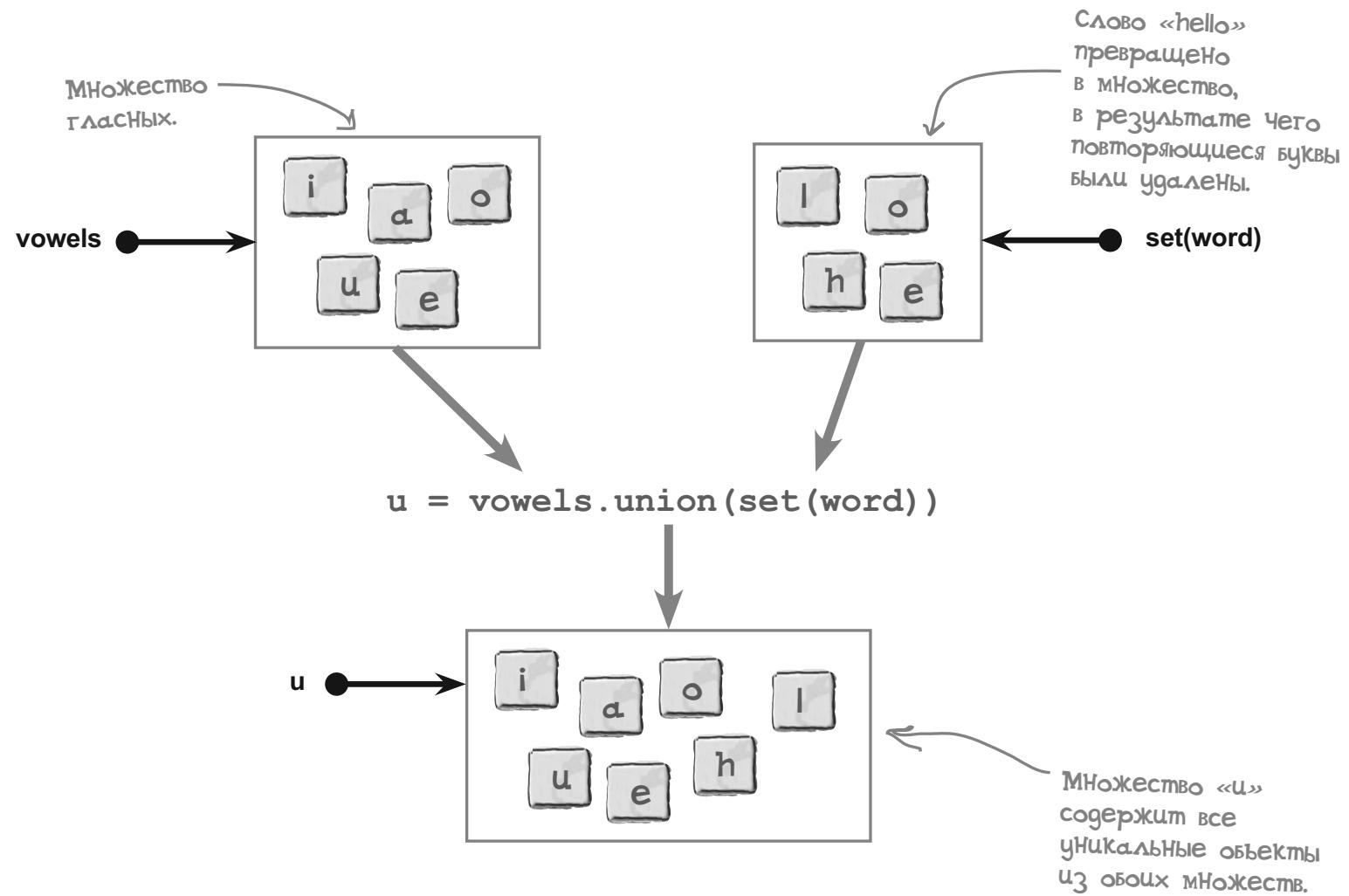
```
>>> vowels = { 'a', 'e', 'e', 'i', 'o', 'u', 'u' }
>>> vowels
{'e', 'u', 'a', 'i', 'o'}
```

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'}
```

Операции с множествами: объединение

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

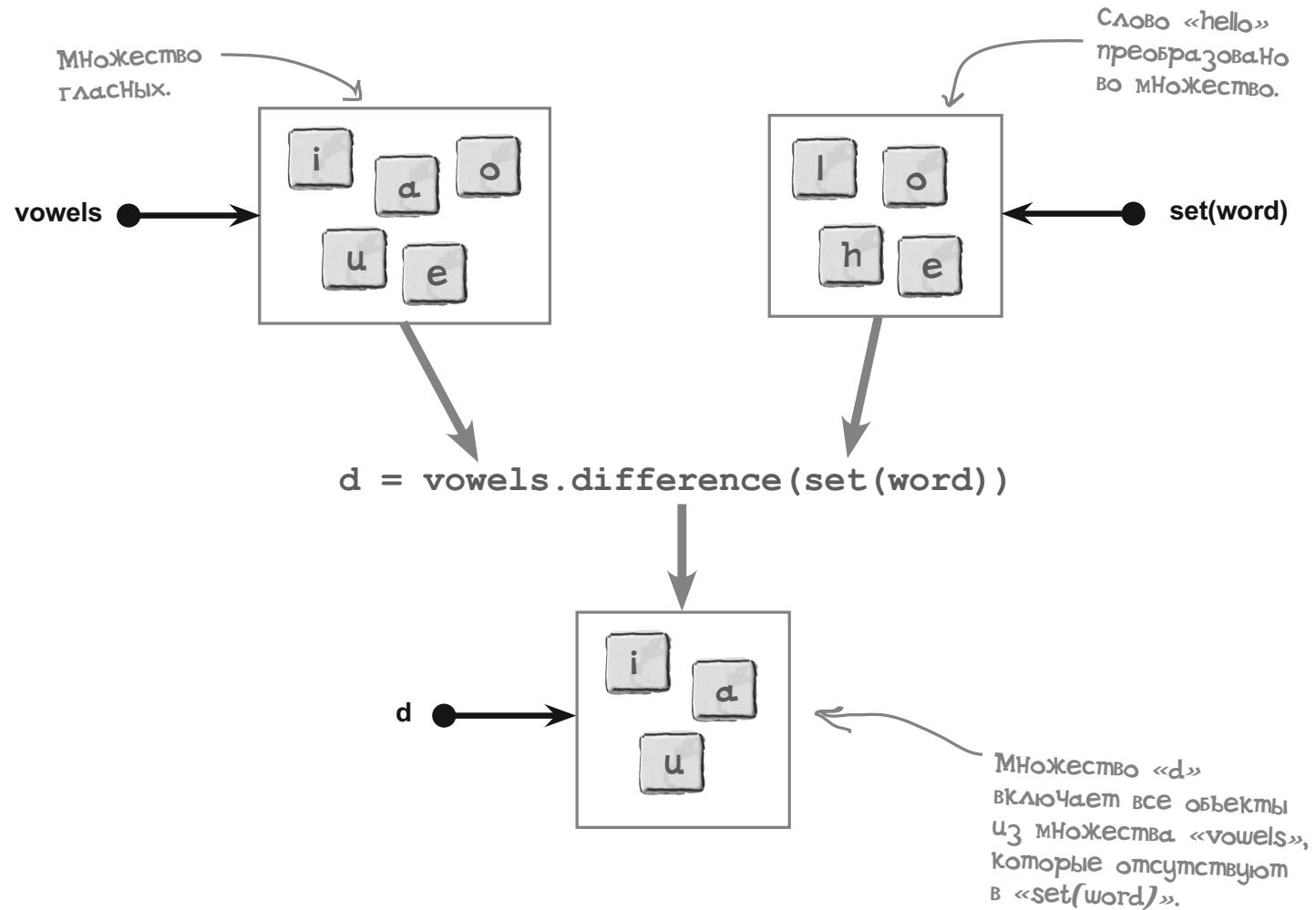


```
>>> u_list = sorted(list(u))
>>> u_list
['a', 'e', 'h', 'i', 'l', 'o', 'u']
```

Операции с множествами: разница

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

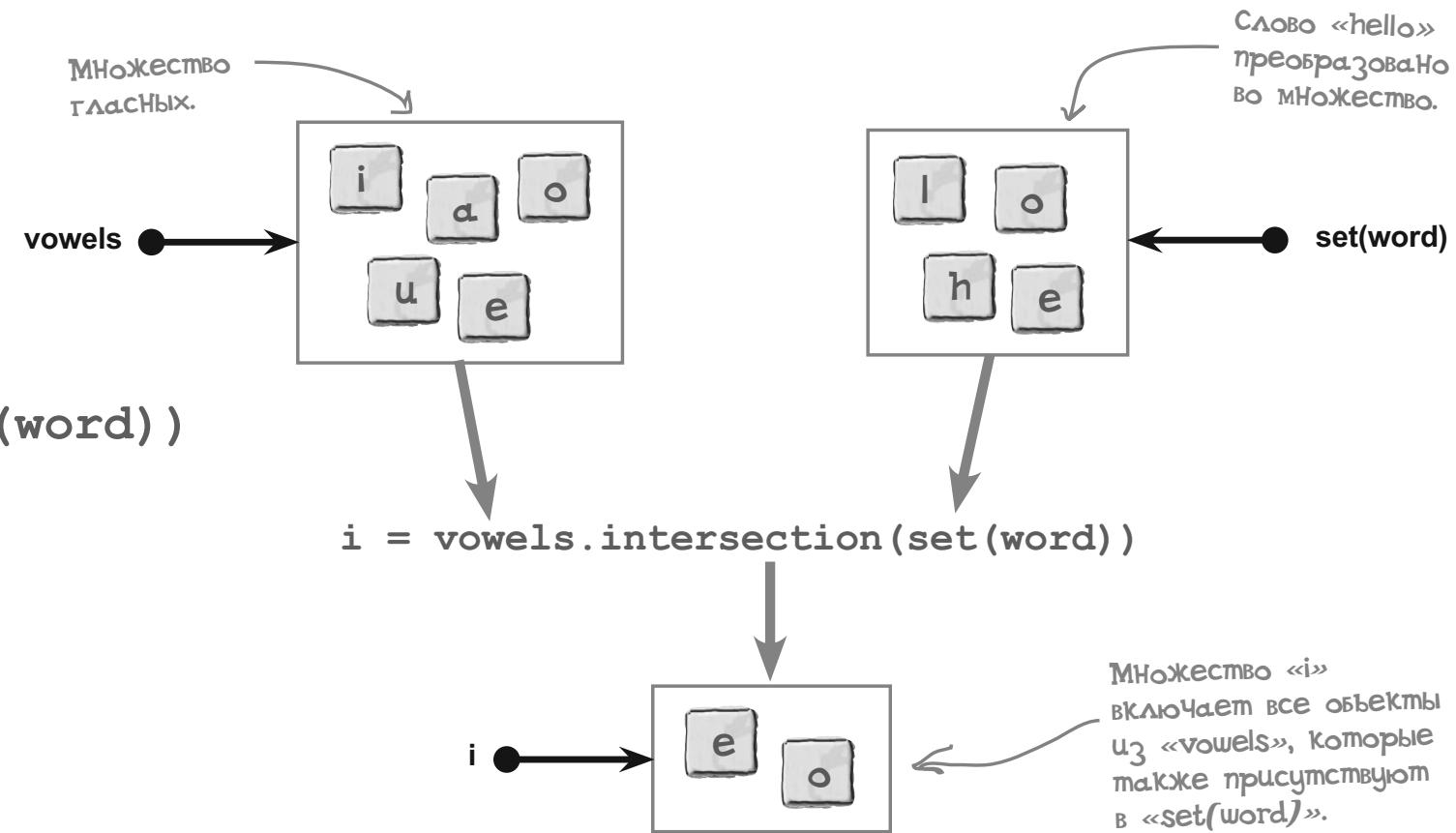
```
>>> d = vowels.difference(set(word))
>>> d
{'u', 'i', 'a'}
```



Операции с множествами: пересечение

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

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



Кортеж (неизменяемый список)

```

    Ничего нового.  

    Создан список  

    гласных.
    Встроенная  

    функция «type»  

    сообщает тип  

    объекта.

    >>> 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.

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

    Круглые скобки показывают,  

    что это кортеж.
  
```

Кортеж (неизменяемый список)

```
>>> vowels2[2] = 'I'  
Traceback (most recent call last):  
  File "<pyshell#16>", line 1, in <module>  
    vowels2[2] = 'I'  
TypeError: 'tuple' object does not support item assignment  
>>> vowels2  
('a', 'e', 'i', 'o', 'u')
```

Интерпретатор
громко возмущается
в ответ на попытку
изменить кортеж.

Ничего не изменилось, потому
что кортежи неизменяемы.

Как создать пустые структуры

Пустой список.

```
>>> l = list()
>>> l
[]
```

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

```
>>> l = [ 1, 2, 3 ]
>>> l
[1, 2, 3]
```

Пустой словарь.

```
>>> d = dict()
>>> d
{}
```

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

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

Как создать пустые структуры

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

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

Пустое
множество.

Пустой
кортеж

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

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()».



ПРЕЗИДЕНТСКАЯ
АКАДЕМИЯ

СПАСИБО ЗА ВНИМАНИЕ!

Москва, 2022

РАНХиГС

2023