

# **Python в DS**

## **Лекция 3: Основы Python**

**Utkin Ilya 20.02.2022**

# Организационные моменты

- Не понимаешь? - спроси.
- Не знаешь? - погугли, не удалось найти - спроси!
- Нагуглил решение? - попробуй повторить сам или хотя бы разобраться!
- Чат в телеграме: <https://t.me/+QotOcGkRqXUyMzhi>
- [Github](#) ([GitMan.pdf](#) для работы с репозиторием)
- “Не ошибается только тот, кто ничего не делает”

# План занятия

- Вспомним что было
- Списки
- Словари
- Циклы. Конструкции if - else

**КАНОТ**

# Типы данных. Строки (str)

- - **последовательность** СИМВОЛОВ для записи текстовой информации.
- **Неизменяемые**
- **Есть много методов, учить не нужно, гуглите!!**

```
s = "строка1"  
print("Для строки {}, адрес: {}".format(s, id(s)))  
s = s.replace("с", "С")  
print("Для строки {}, адрес: {}".format(s, id(s)))
```

```
Для строки строка1, адрес: 140368660461024  
Для строки Строка1, адрес: 140368660410768
```

**Идем в ноутбук!**



# Challenge

# Типы данных. Булевые

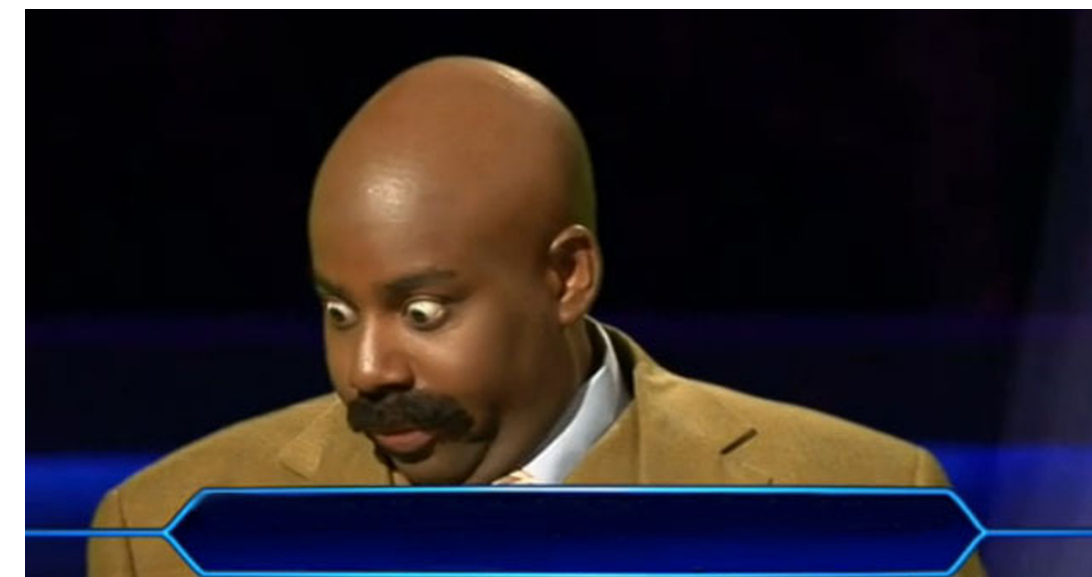
- Логический тип, представлен True, False значениями. По своей сути это integer с значениями 1 и 0

```
a = True
b = True
print(id(a))
print(id(b))
```

```
94842126869248
94842126869248
```

```
print(True == False)
print(True == True)
print(False == False)
```

```
False
True
True
```



```
if a is None:
    a = 10
```

Идем писать код

```
a = 5
b = 5
a is b
```

```
True
```

```
a = 1000
b = 1000
a is b
```

```
False
```

Почему???

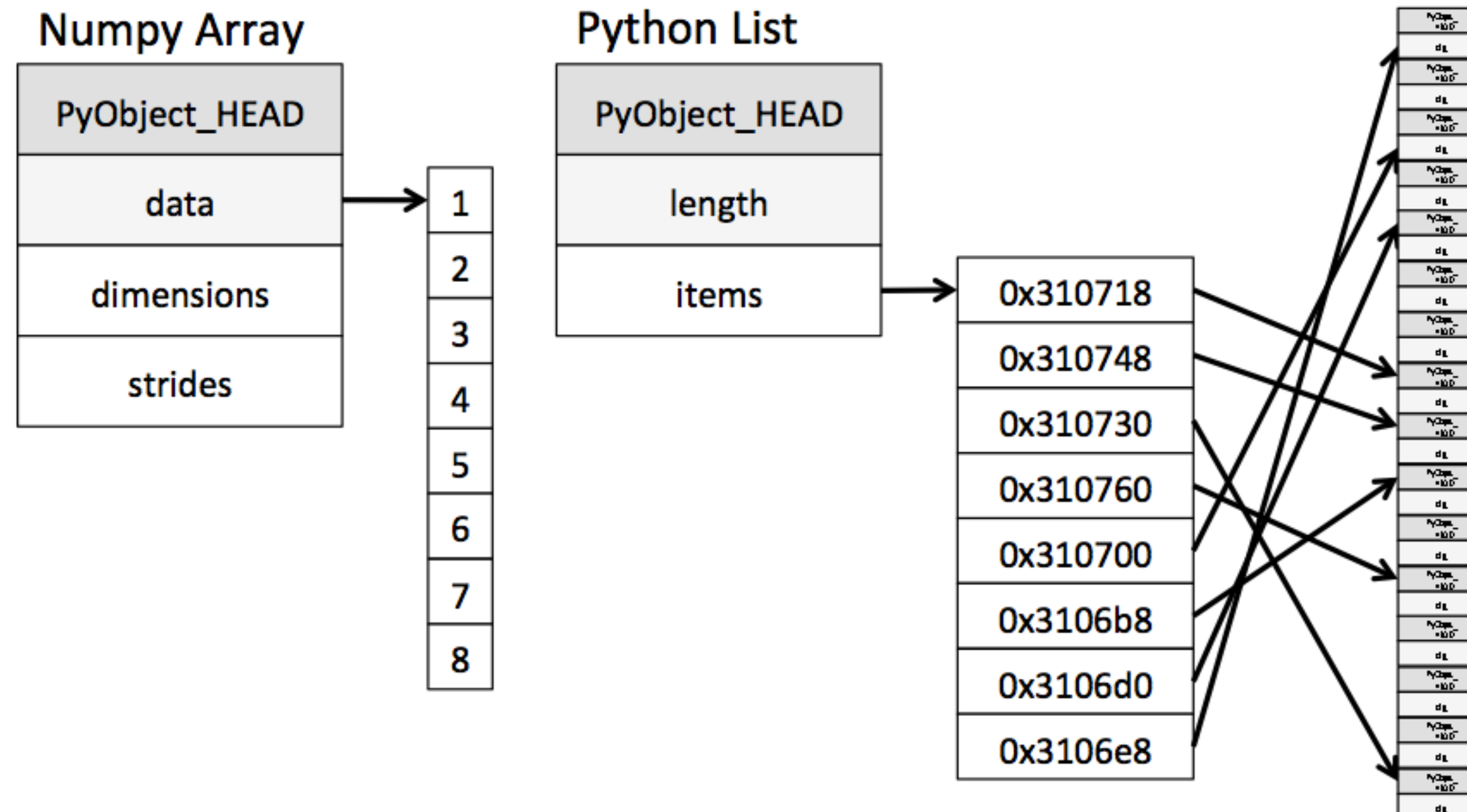




# Типы данных. Массивы (list)

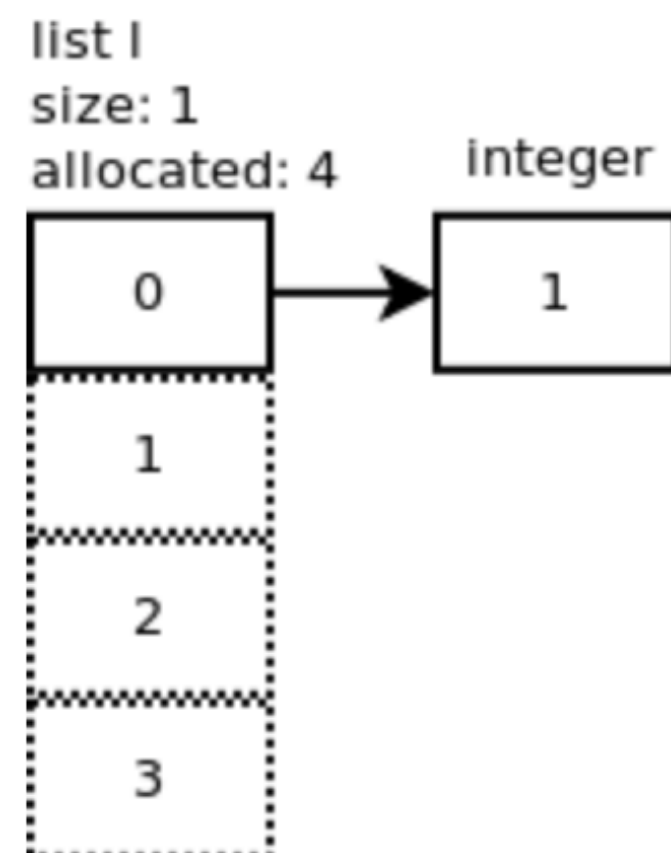


- Массивов на самом деле нет в python. Есть списки.
- *Список в python - динамический массив указателей.*

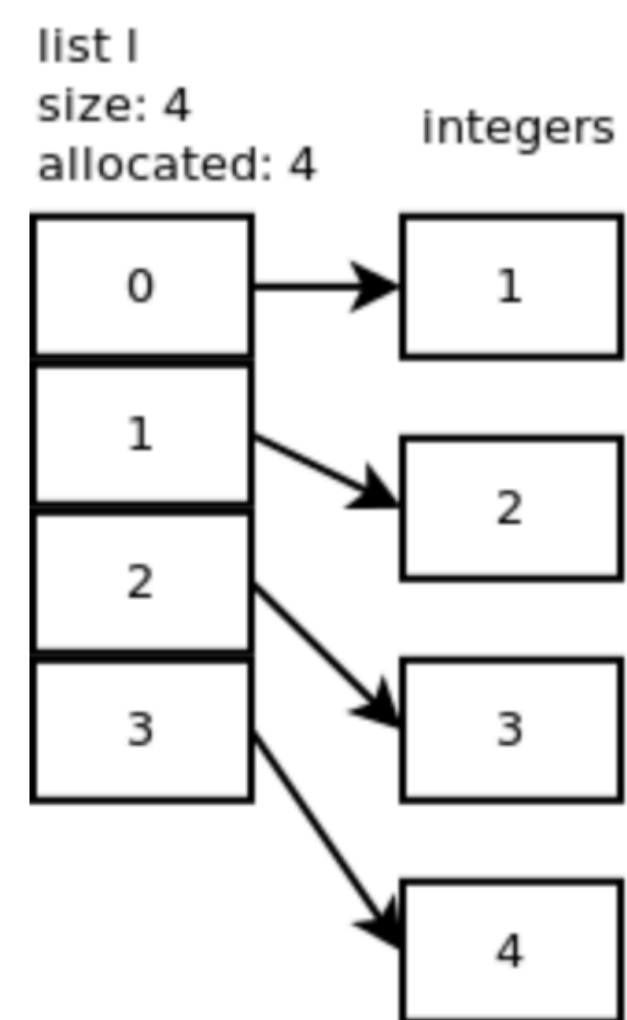




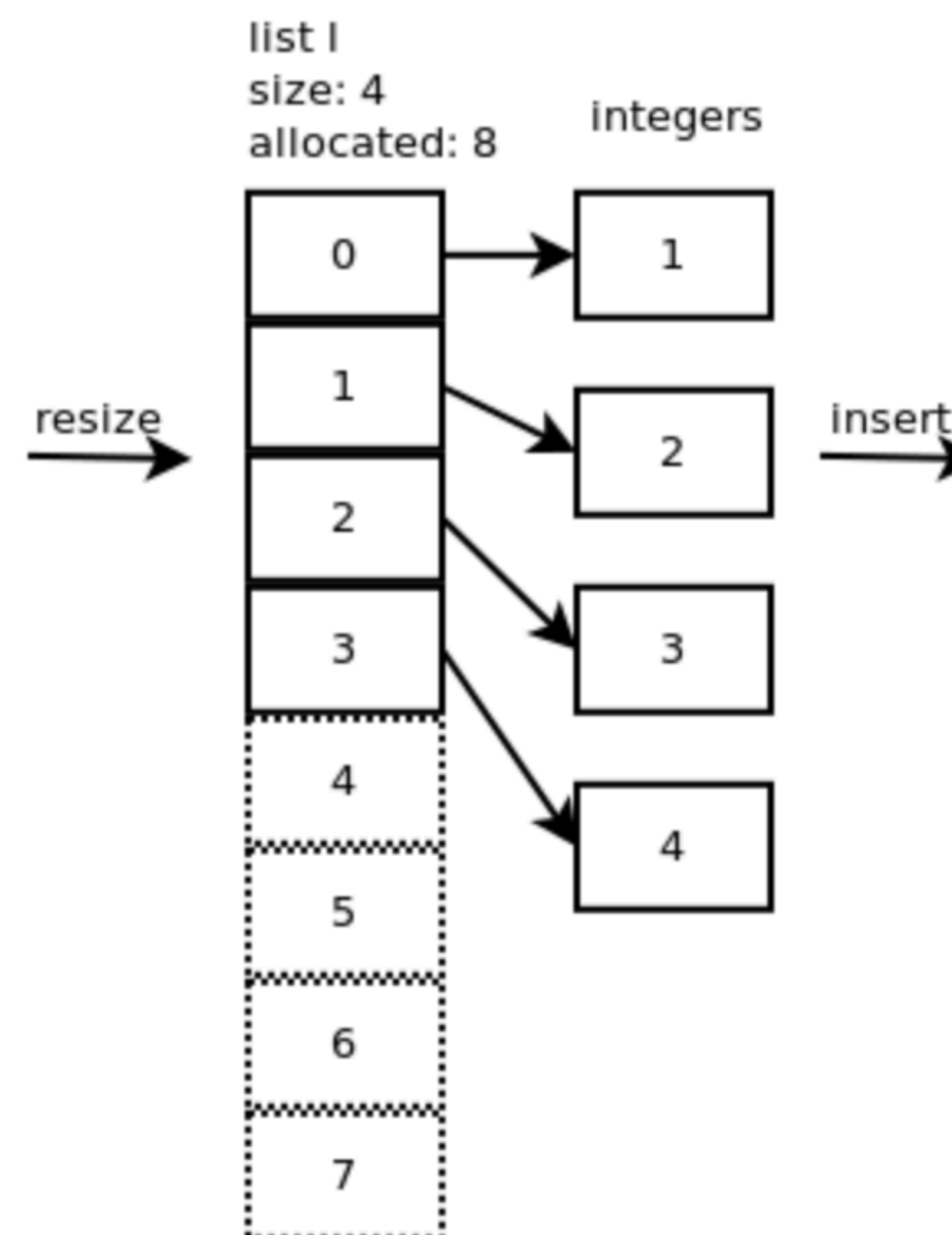
# Типы данных. Массивы



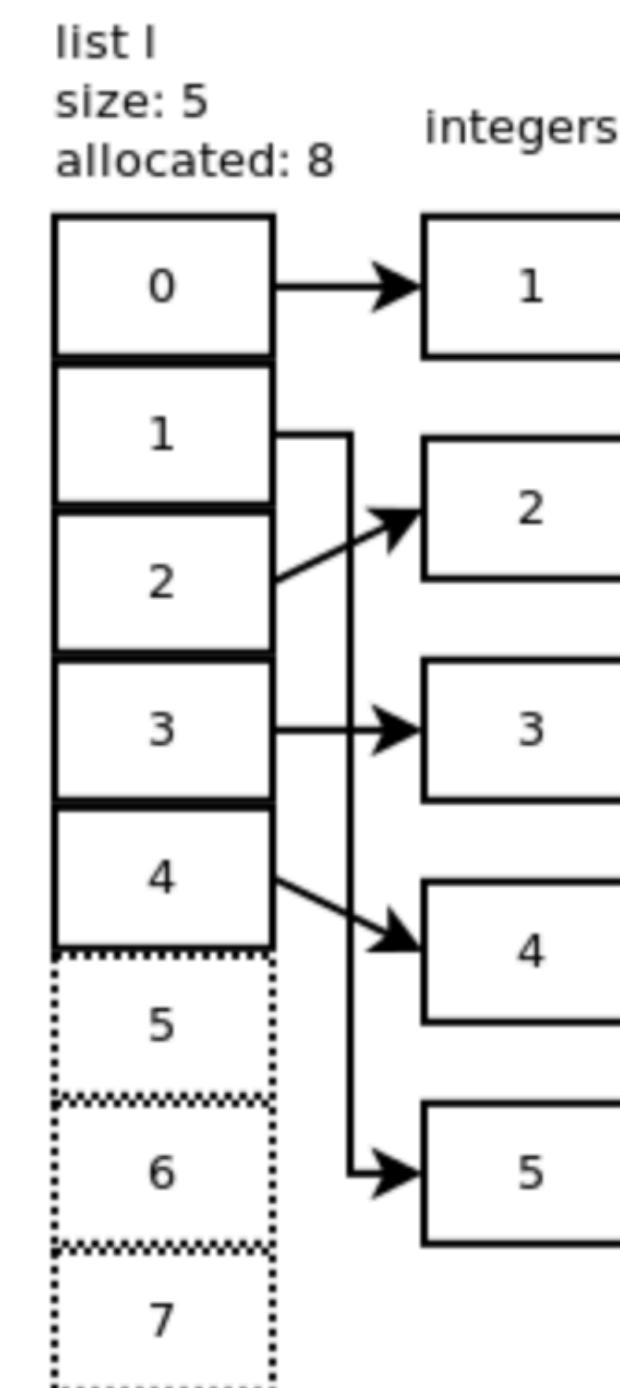
Список с одним  
элементом



Заполнили всю выделенную  
память



Добавление еще одного  
элемента в конец



Добавление еще одного  
элемента в 1

# Типы данных. Массивы

```
a = []  
a = list()  
a = [1, 2, 'яблоко']  
a = [i for i in range(10)]  
a = [[i for i in range(10)] for _ in range(5)]  
a = [0] * 5
```

```
a = [0, 1, 2]  
print(a.pop())  
print(a)
```

```
2  
[0, 1]
```

```
a = [0, 1, 2, 3]  
b = [4, 5, 6]  
a.append(b)  
print(a)
```

```
[0, 1, 2, 3, [4, 5, 6]]
```

```
a = [0, 1, 2, 3]  
b = [4, 5, 6]  
a += b  
print(a)
```

```
[0, 1, 2, 3, 4, 5, 6]
```

Идем писать код

[Подробнее](#)



# Типы данных. Tuple

Кортеж - неизменяемый! набор упорядоченных данных  
Неизменяемость позволяет быть ключом словаря (рассмотрим позже)

Но неизменяемый сам tuple, а не содержащиеся в нем элементы

Зачем они нужны если есть списки?

```
a = tuple()
print(a)
a = (1, 2)
print(a)
```

```
()
(1, 2)
```

```
a = 3
b = 5
a, b = b, a
print(a)
print(b)
```

```
5
3
```

```
t = tuple([[1,2,3], ['a','b','c']])
```

executed in 11ms, finished 12:13:05 2018-09-23

```
type(t)
```

executed in 4ms, finished 12:13:06 2018-09-23

tuple

```
t
```

executed in 4ms, finished 12:13:06 2018-09-23

```
([1, 2, 3], ['a', 'b', 'c'])
```

```
t[0].append(12)
```

executed in 11ms, finished 12:13:07 2018-09-23

```
t
```

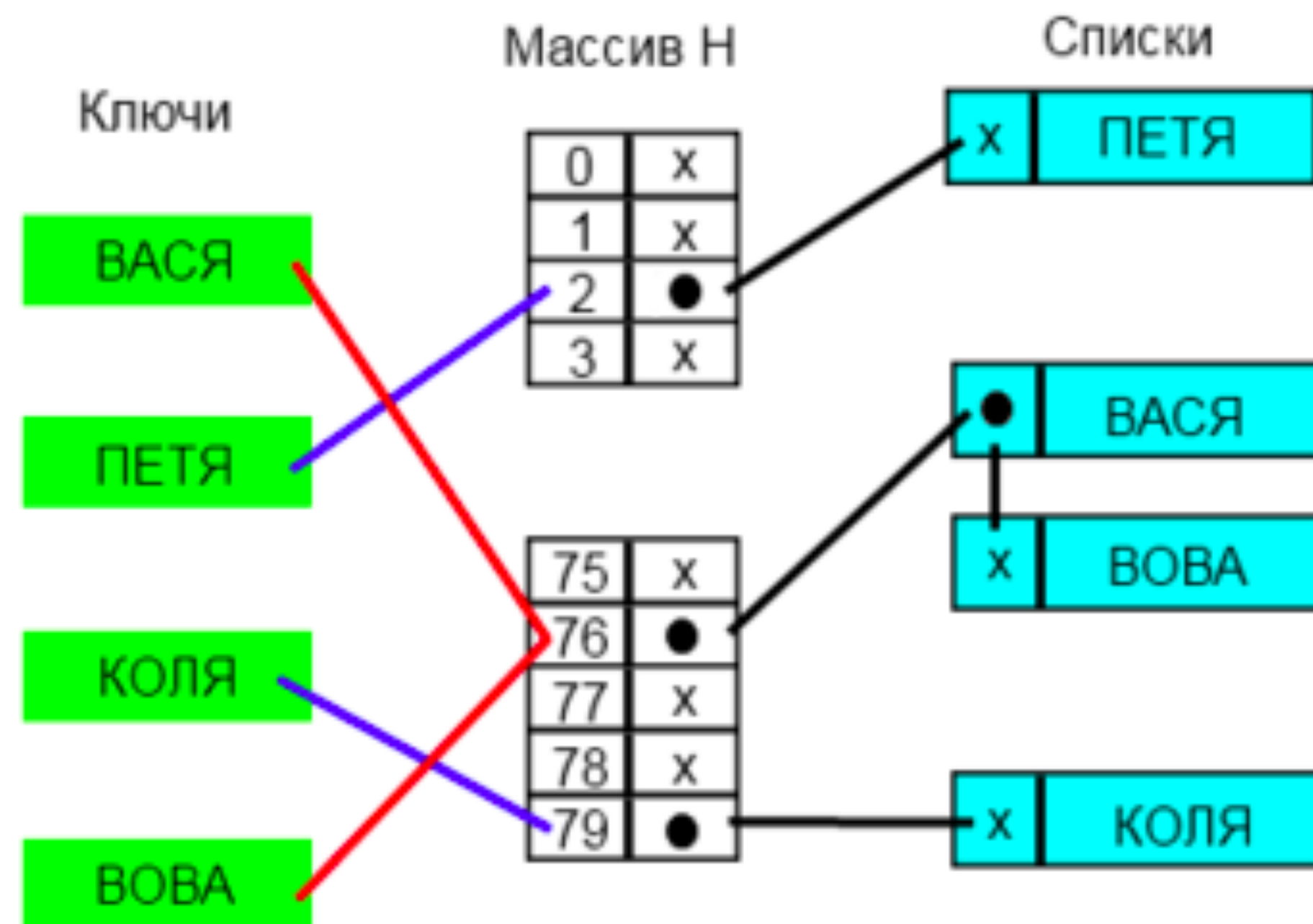
executed in 9ms, finished 12:13:07 2018-09-23

```
([1, 2, 3, 12], ['a', 'b', 'c'])
```

# Типы данных. Словари. Хэширование

По сути хеширование - это отображения множества ключей на множество значений хеш-функции

**Хеш-таблица** — это структура данных, реализующая интерфейс ассоциативного массива, а именно, она позволяет хранить пары (ключ, значение) и выполнять три операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу.



Разрешение коллизий с помощью цепочек

Существует два основных вида хеш-таблиц: с *цепочками* и *открытой адресацией*. Хеш-таблица содержит некоторый массив  $H$ , элементы которого есть пары (хеш-таблица с открытой адресацией) или списки пар (хеш-таблица с цепочками).

Выполнение операции в хеш-таблице начинается с **вычисления хеш-функции от ключа. Хеш-код  $i=h(key)$**

играет роль индекса в массиве  $H$ , а зная индекс, мы можем выполнить требующуюся операцию (добавление, удаление или поиск).

Количество коллизий зависит от хеш-функции; чем лучше используемая хеш-функция, тем меньше вероятность их возникновения.



Пример хеш-таблицы с открытой адресацией и линейным пробированием.



# Типы данных. Словари (dict)

Ключи - только неизменяемые объекты.

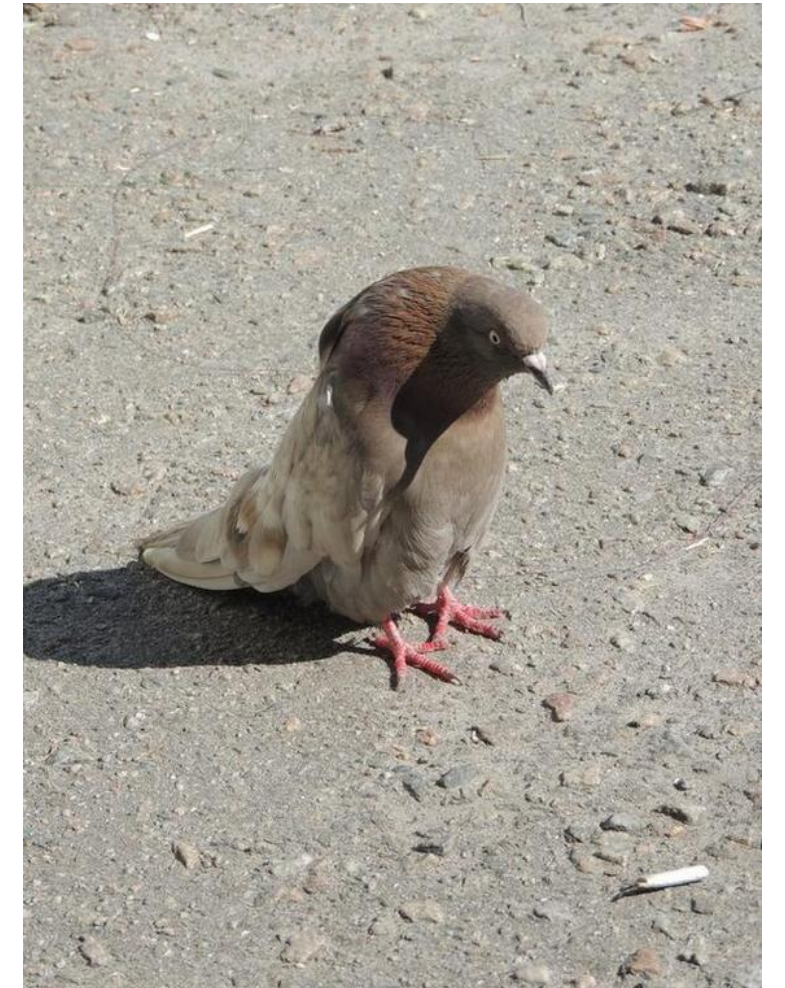
```
d = dict(key="value")
d = {"key": "value"}
d = dict([("key_1", "value_1"), ("key_2", "value_2")])
d = dict.fromkeys(['key_1', 'key_2'], "value")
```

```
d[[1, 2]]
```

```
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-58-084c77bd943b> in <module>()
----> 1 d[[1, 2]]
```

```
TypeError: unhashable type: 'list'
```

```
d[(1, 2)] = "э"
```



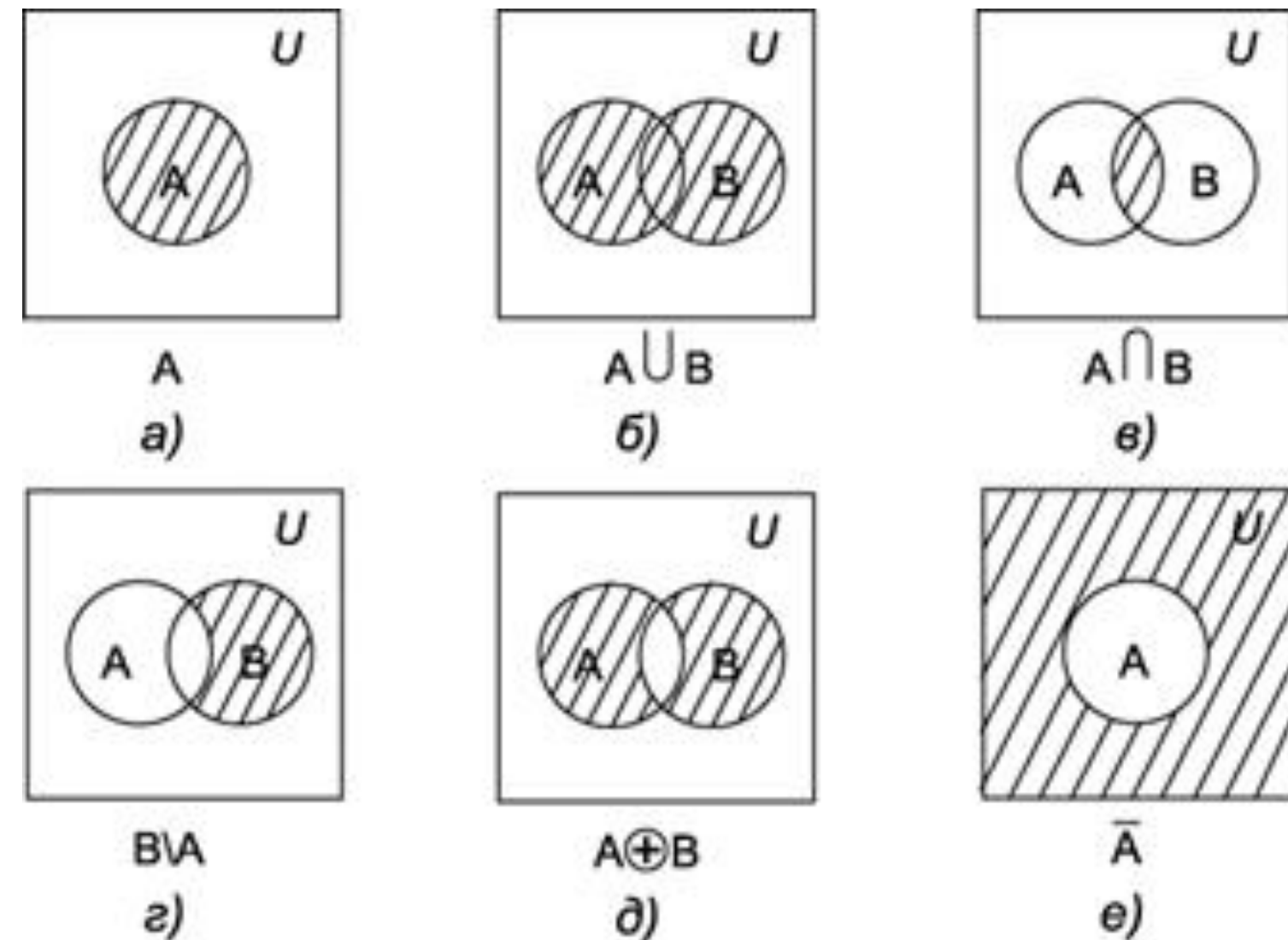
Идем писать код



# Типы данных. Множество (set)

**Множество (математика)** - набор, совокупность каких-либо (вообще говоря любых) объектов — элементов этого множества.

**Множество (python)** - структура данных, содержащая элементы (объекты) в случайном порядке. С точки зрения реализации - это dict, у которого вместо value установлены заглушки.



Идем писать код



# Оставьте обратную связь

<https://forms.gle/W4EsPkB4caFx8UgQA>





# Полезные ссылки и материалы

1. [Раз](#) - типы данных в ПИТОН
2. [Два](#) - Jupyter / collab
3. [Три](#) - Питон в 3 страницах
4. Книги:
  - Марк Лутц. Изучаем python.
  - Лучано. Python к вершинам мастерства
5. Очень советую [курс](#), в дополнение.

