Python в DS

Лекция 1: Вводное занятие. Основы Python

Utkin Ilya 07.02.2022

Организационные моменты

- Не понимаешь? - спроси.
- Не знаешь? - погугли, не удалось найти - спроси!
- Нагуглил решение? - попробуй повторить сам или хотя бы разобраться!
- Мой телеграм: @Anglarp.
- Github (GitMan.pdf для работы с репозиторием)
- “Не ошибается только тот, кто ничего не делает”

План курса

- Python с нуля (типы данных, функции, ООП, мультипроцессинг)
- Анализ данных (библиотеки numpy, pandas, matplotlib, немного математики, немного магии - нейросети)
- Pet project (бот на питоне, распознающий рукописные цифры)

Почему python?

- Низкий порог вхождения
- Широкое применение (от сайтов, десктопа до научных исследований, ~~мобилки~~)
- Сильное коммьюнити (хорошие библиотеки, stackoverflow)
- Высокий спрос
- Быстрая разработка

Hello World на Java

```java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Hello World на Python

```python
print("Hello, World!")
```

Hello World на C++

```cpp
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello, World!";
    return 0;
}
```

Интерпретация vs компиляция

- **Компиляция**

  Исходный код -> низкоуровневый код

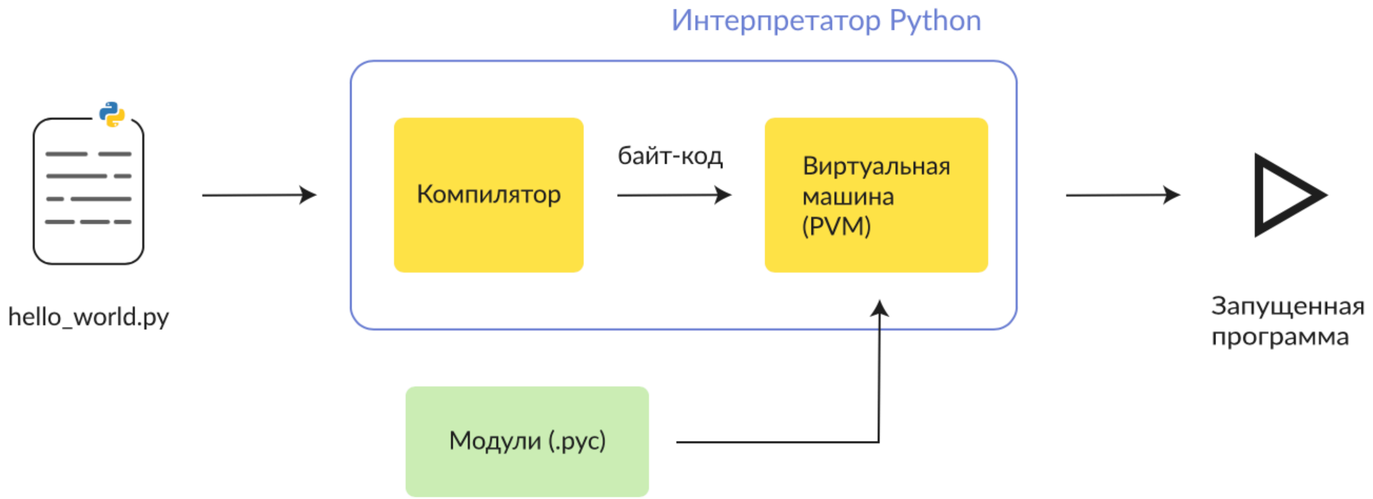
  - C++ -> машинный, может быть выполнен непосредственно процессором.
  - Java -> байт код для Java Virtual Machine (JIT).

- **Интерпретация**

  Программу построчно исполняют интерпретаторы (программы)
  1. Чтение инструкций и анализ
  2. Выполнить инструкцию
  3. Repeat.

CPython интерпретатор

- Эталонная реализация языка Python
- Написан на C
- Open source
- Комплирующего типа



  - **Альтернатива - PyPy:**

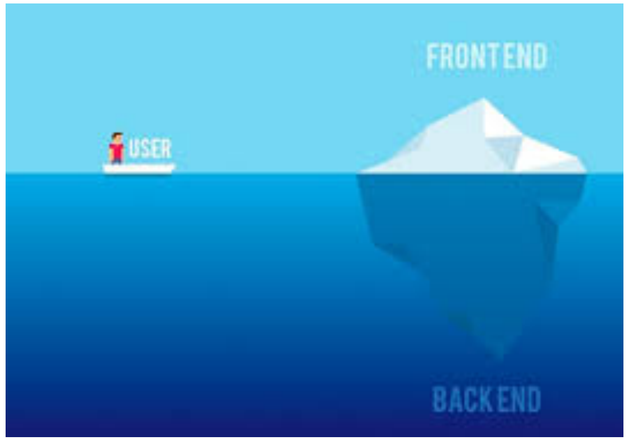
    - налету транслирует некоторые инструкции в машинный код (JIT)

    - быстрее CPython

    - ест больше памяти

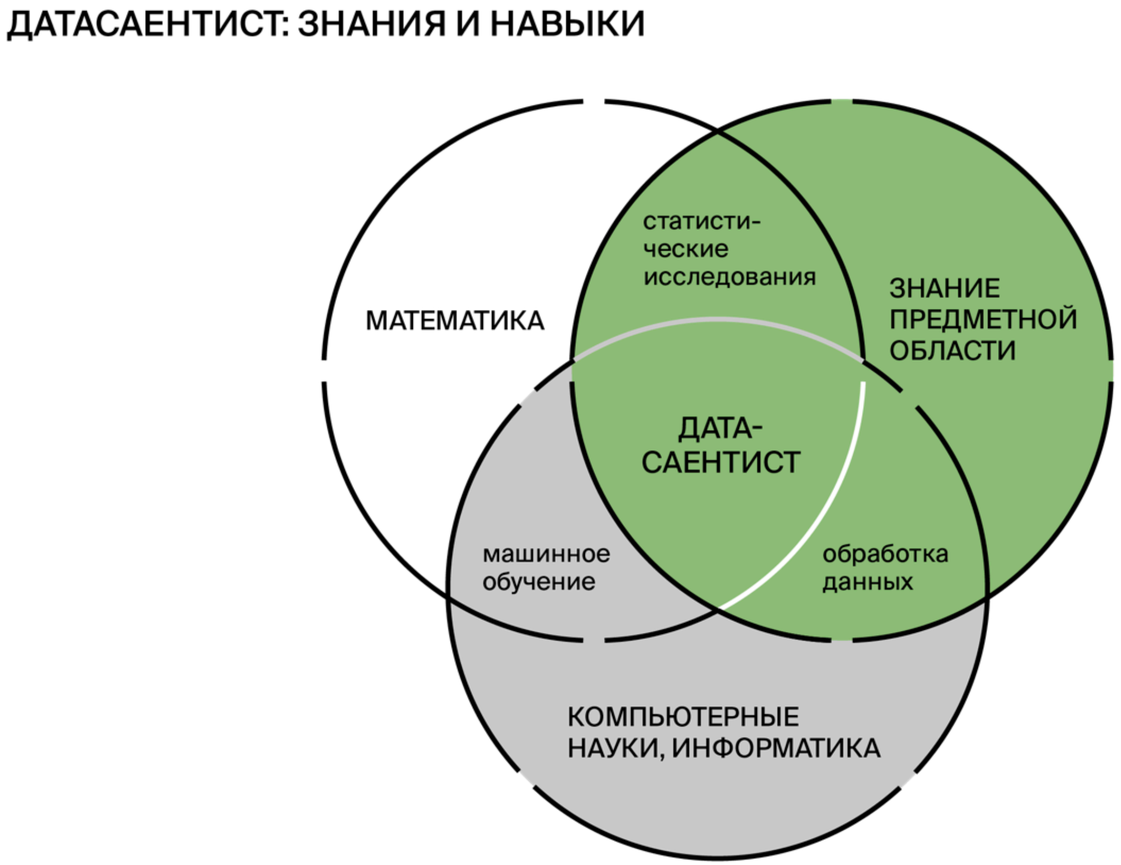
Применение Python

- Web разработка: Flask, Django. (Instagram, BitTorrent до 6 версии)
- Автоматизация, тестирование, DevOps. (Практически везде)
- Наука, DS, AI: Pytorch, sklearn, Tensorflow. (NASA, CERN)



Data Science

- применение научных методов при работе с данными



- Обрабатывает массивы данных, находит в них новые связи и закономерности, используя алгоритмы машинного обучения, и строит модели.
- Модель — это алгоритм, который можно использовать для решения бизнес-задач.
- Работает с табличными данными, картинками, графами, изображениями, звуком, видео.

ML области

- Компьютерное зрение (**C**omputer **V**ision)
  - Self driving cars (https://www.youtube.com/watch?v=1Yc1v-LXI-0)
  - face detection (https://www.youtube.com/watch?v=PL3xJErjEgU)
  - style transfer (https://t.me/face2comicsbot)

- Speech2Text (Siri)

- Обработка языка (**N**atural **L**anguage **P**rocessing) (спам фильтр, чат боты, Siri, суммаризация)
  - ruDalle (СБЕР: https://rudalle.ru/demo)

- Прогнозирование (цены, погоду, курс, все - что можно)

Python

- Запуск скриптов python3 hello_world.py. Попробуем вместе в терминале.

- Интерактивный сеанс интерпретатора python3.

```
$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Jupyter notebook / Google colab. (https://colab.research.google.com)

PEP8. Или как надо оформлять код

“Код читается намного больше раз чем пишется”

- 4 пробела на каждый вид отступа (не табуляции)

```
def func1():
    def func2():
        a = 1
```

- Ограничьте длину строки в 78 символов
- Кодировка python3 - UTF8
- ….

  Ознакомиться можно тут

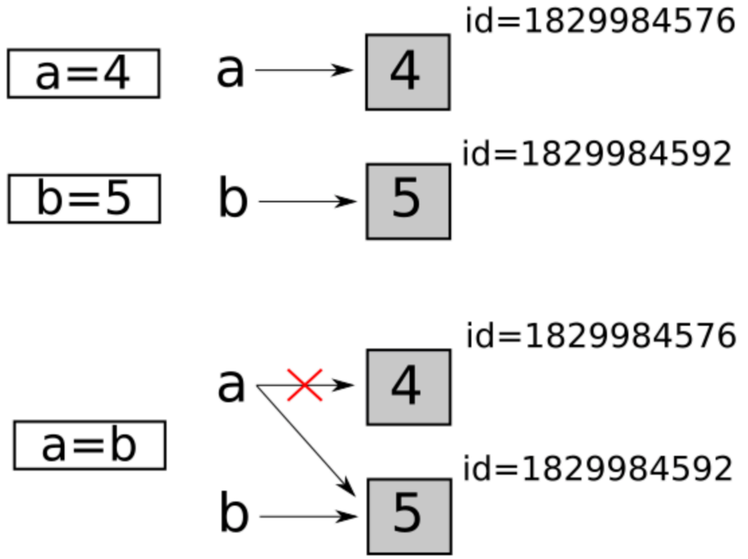
Типы данных. Числа (int)

- Integer (целые числа). Ограничены только доступной памятью
- Float (с плавающей точкой). Ограниченная точность.
- Complex (мнимые числа)

```
a = 5
b = 5.0
print("Type of a is {}".format(type(a)))
print("Type of b is {}".format(type(b)))
```

```
Type of a is <class 'int'>
Type of b is <class 'float'>
```

Идем в ноутбук!



Challenge

Типы данных. Строки (str)

- - последовательность символов. Можем использовать ‘строка’ или “строка”.
- Неизменяемые

```
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

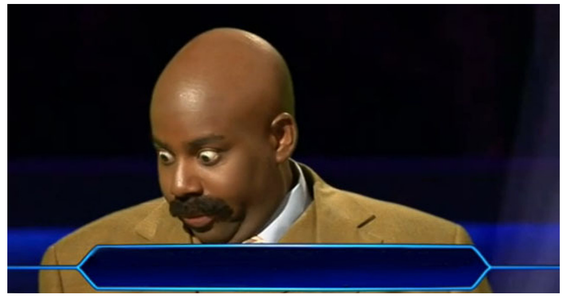
- Логический тип, представлен True, False значениями. По своей сути это integer с значениями 1 и 0

```
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 в ноутбук!

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

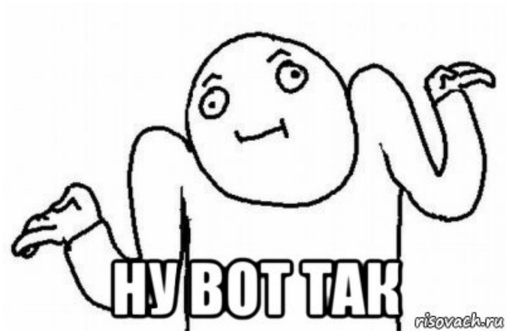
```
True
```

```
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.*

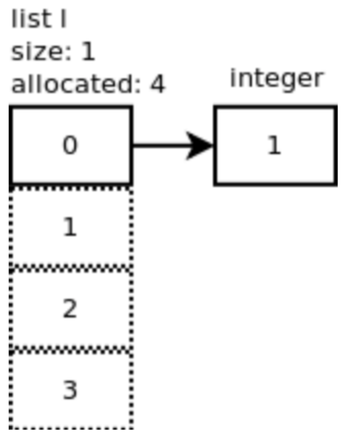
```
typedef struct {
    PyObject_VAR_HEAD
    PyObject **ob_item;
    Py_ssize_t allocated;
} PyListObject;
```

Структура списка

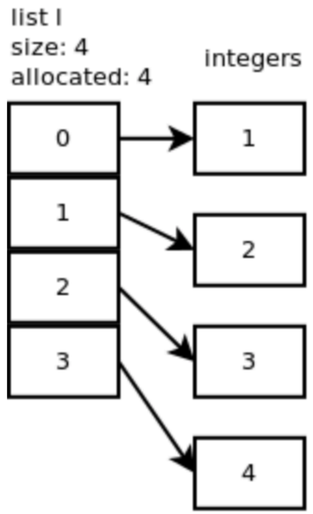
Инициализация
= выделение памяти

```
/*
size – размер списка
*/
PyList_New(Py_ssize_t size)
{
    // Вычисляется реальный размер необходимой памяти
    nbytes = size * sizeof(PyObject *);

    // Инициализируется ob_item
    if (size <= 0)
        op->ob_item = NULL;
    else {
        op->ob_item = (PyObject **) PyMem_MALLOC(nbytes);
        memset(op->ob_item, 0, nbytes);
    }

    // Сохраняется количество выделенных ячеек
    op->allocated = size;

    return (PyObject *) op;
}
```

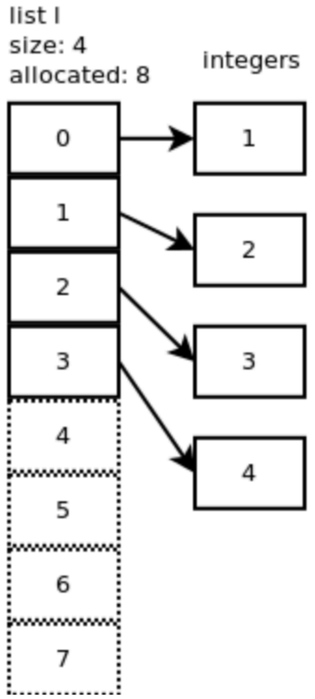
Типы данных. Массивы



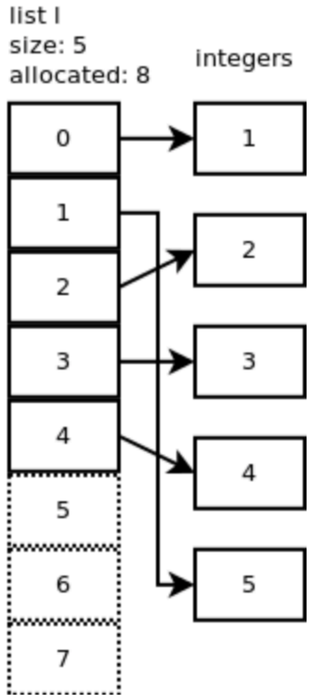
Список с одним элементом



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 в ноутбук!

[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

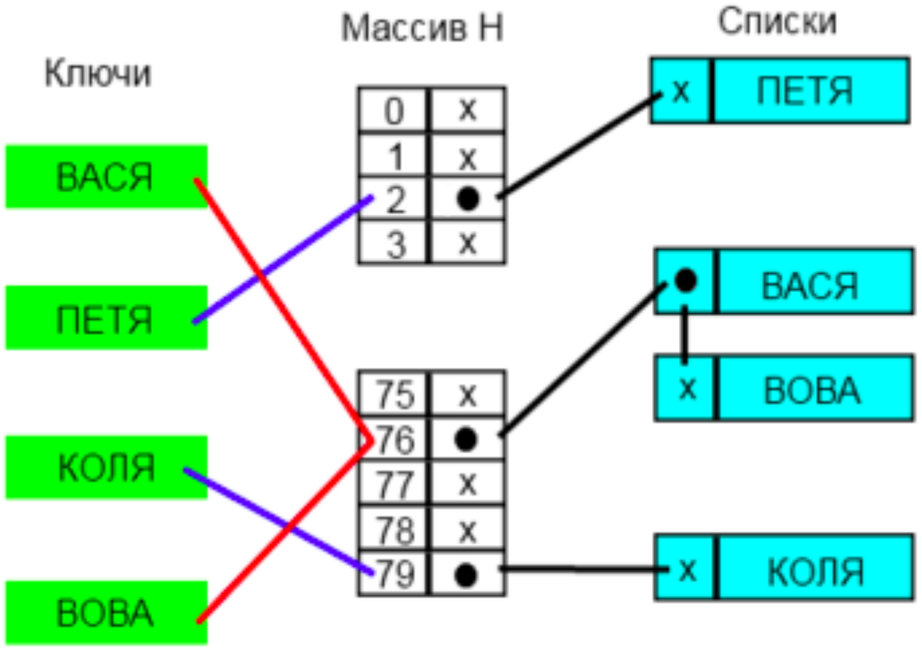
```
t
```
executed in 9ms, finished 12:13:07 2018-09-23

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

Типы данных. Словари. Хэширование

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

Хеш-табли́ца — 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.



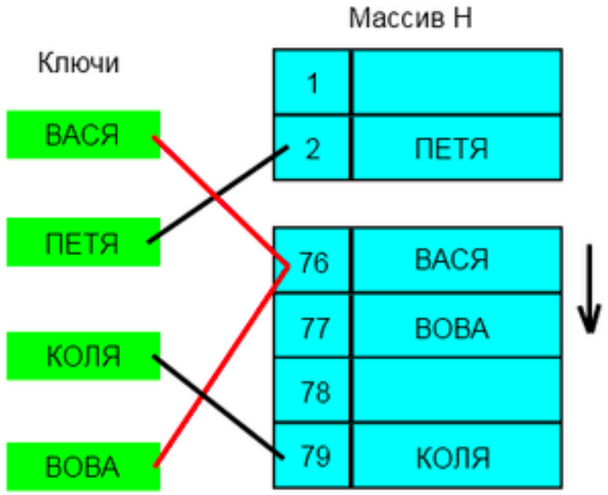
Разрешение коллизий с помощью цепочек

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)

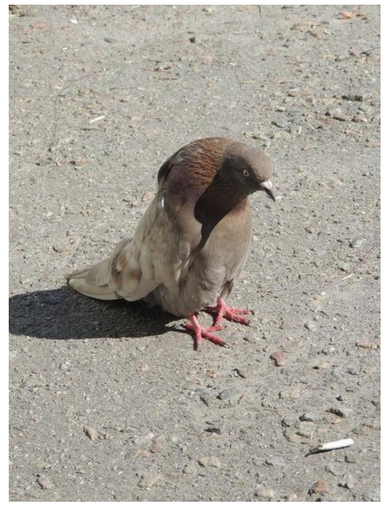
Ключи - только неизменяемые объекты.

```
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"
```

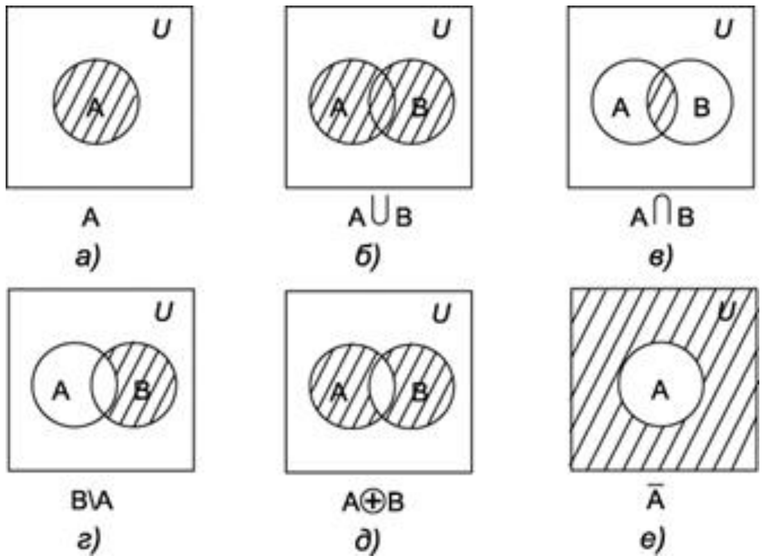


Идем в ноутбук!

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 в ноутбук!

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