

**ТЕХНОАТОМ**

# **Смешанное занятие № 4**


## **Введение в Python**

Елена Ставровская





## Цель занятия

- Повторить синтаксис языка
  - Разобрать «фишечки»
  - Познакомиться с библиотекой Pandas
- 



# Содержание занятия

1. Что такое Python;
2. Синтаксис и стиль;
3. Условный оператор;
4. Циклы;
5. Функции;
6. Фишечки: lambda функции и списочные сокращения;
7. Библиотека Pandas;



# Что такое Python

Python – высокоуровневый, транслируемый в байт-код, объектно-ориентированный, регистро-зависимый язык.

Python не требует явного объявления переменных, поддерживает многопоточные вычисления.

# IDE для работы с Python

- Visual Studio + Visual Studio Code;
- - PyCharm;
- - Spyder;
- - Jupyter Notebook + JupyterLab;
- - Domino.



# Python: Память

Python сам управляет памятью и заморачиваться о разного рода процедурах с аллоцированием памяти и её очисткой здесь не надо.

## Си

```
char *src = malloc(len);
char *dst = malloc(len);

src = scanf("%s", len);
dst = memcpy(dst, src, len);
...
free(src);
free(dst);
```

vs

## Python

```
def foo():
    src = input()
    dst = src
```

Но иногда вы можете и помочь.

Например, когда вы выталкиваете элемент из списка, `l.pop()` внутри вызывает `list_resize` и в случае, если новый размер меньше половины выделенной памяти — то список сжимается и памяти тратится меньше.

# Python: стандарты стиля и синтаксиса

Код читается намного больше раз, чем пишется. Поэтому пишите аккуратно.

Например, по [PEP8](#) от Google.

В этом документе в т.ч. затрагиваются следующие основные пункты:

- Кодировка исходного кода;
- Выделение блоков, сколько ставить пробелов;
- Именованние переменных;
- Наследование и доступность классов и методов;
- Импорт модулей.



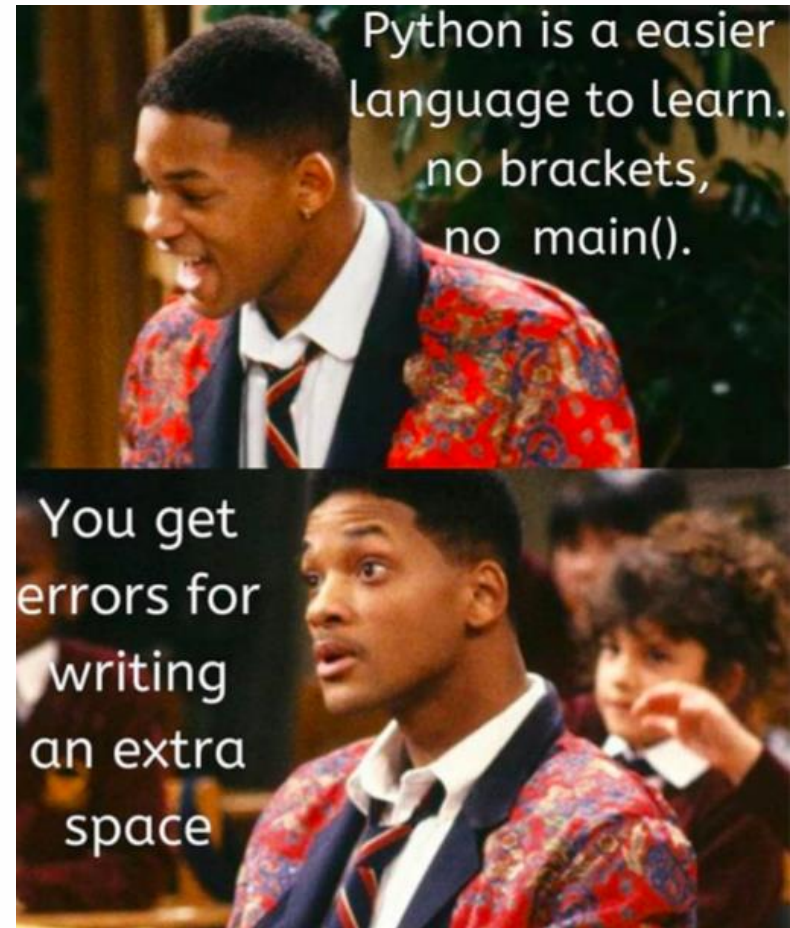
Автор языка Python, Гвидо ван Россум, на основе рекомендаций которого был создан документ PEP 8

# Python: Синтаксис

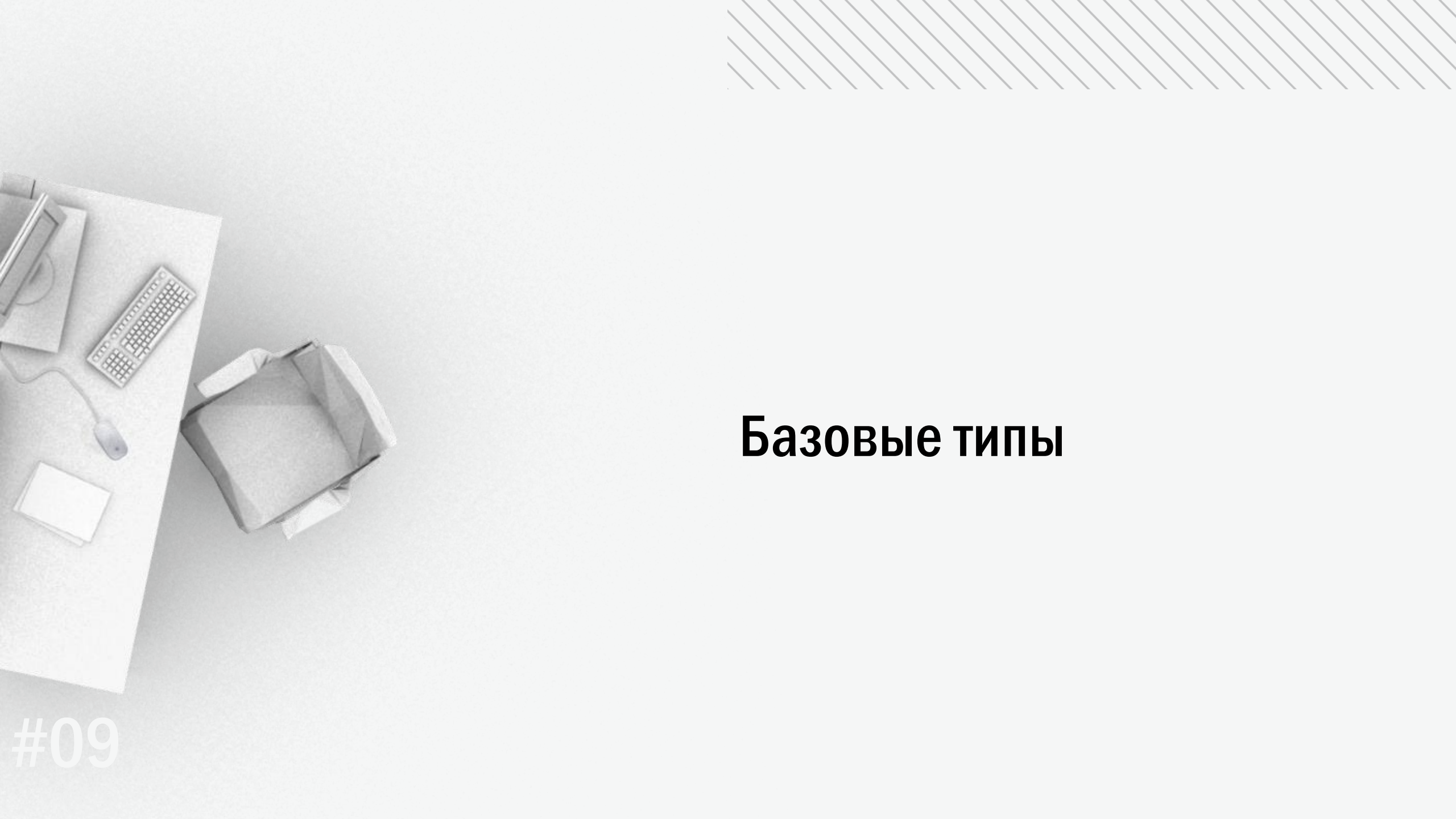
Python не содержит операторных скобок, как Си, паскаль или php, например.

Вместо этого блоки надо здесь выделять табуляцией или пробелами.

Лишний/недостающий пробел, и вы получаете ошибку







# Базовые типы

# Python: Типы объектов

Все в python объекты

```
1 a = 1
2 print(isinstance(a, object))
```

True

```
1 fn = lambda x: x**2
2 print(isinstance(fn, object))
```

True

# Python: Типы объектов

В Python существуют изменяемые и неизменяемые объекты.

Тип	Неизменяемый?
int	Да
float	Да
bool	Да
complex	Да
tuple	Да
frozenset	Да
str	Да
bytearray	Нет
list	Нет
set	Нет
dict	Нет

## • Числовые типы

Int – целый

Float – с плавающей точкой

Числовые типы являются неизменяемыми

1	<code>a = 1</code>
2	<code>id(a)</code>

1658154048

1	<code>b = 0.9</code>
2	<code>id(b)</code>

1891969443424

1	<code>a += 1</code>
2	<code>id(a)</code>

1658154080

1	<code>b = b + 1</code>
2	<code>id(b)</code>

1891969443376

# Python: [Строки] str

Строки (str) обособляются двойными или одинарными кавычками. Мультистрочные выражения нужно выделять тремя кавычками.

Объект str представлен в виде массива символов и является итерируемым. Объект str сохраняет длину, хэш от содержания и флаг, определяющий, хранится ли строка в кэше.

Можете заполнять str любой длиной, на сколько вам хватит памяти.

```
1. typedef struct {  
2.     PyObject_VAR_HEAD  
3.     long ob_shash; # хэш от строки  
4.     int ob_sstate; # находится ли в кэше?  
5.     char ob_sval[1]; # содержимое строки + нулевой байт  
6. } PyStringObject;
```

C

# Python: [Логический тип] bool

Логический тип был добавлен в Python 2.3, и он просто расширяет тип int.

True и False существовали до какого-то времени в v2.x как изменяемые объекты.

```
1.bool_repr(PyBoolObject *self){
2.    ...
3.    s = true_str ?
4.        true_str :
5.        (true_str =PyString_InternFromString("True"));
6.    ...
7.    s = false_str ?
8.        false_str :
9.        (false_str = PyString_InternFromString("False"));
10.    ...
11.    return s;
12.}
```

---

# Python: [Ничего] None

Объект None является единственным значением специального типа `NoneType`.

Используется для того чтобы показать отсутствие значения

```
1. type(None)
```

При преобразовании к типу `bool` всегда получается `false`.

# Python: [Списки] list

Списки (list) в Python – упорядоченные изменяемые коллекции объектов произвольных типов. Итерируемый тип.

Объект `ob_item` – список указателей на элементы списка и `allocated` – выделенный объём в памяти.

Память для списков выделяется пропорционально размеру списка.

Кроме того, алоцированный объём памяти также пересчитывается, если количество элементов меньше половины выделенных слотов памяти.

```
1. typedef struct {  
2.     PyObject_VAR_HEAD  
3.     PyObject **ob_item;  
4.     Py_ssize_t allocated;  
5. } PyListObject;
```



# Python: [Кортежи] tuple

Кортеж (tuple) представлен в виде массива из указателей и является неизменяемым итерируемым объектом.

```
1. a = (1, 2, 3)
```

Кортежи занимают меньше памяти, чем списки.

```
1. t = (1, 2, 3)
2. sys.getsizeof(t)
```

VS

```
1. l = [1, 2, 3]
2. sys.getsizeof(l)
```

```
1. typedef struct {
2.     PyObject_VAR_HEAD
3.     PyObject *ob_item[1];
4. } PyTupleObject;
```

C

# Python: [Словари] dict

В словарях (dict) для хранения своих данных используются хеш-таблицы, которые по сути являются всего лишь некоторыми массивами. Словари являются итерируемыми.

Хеш-таблица представлена набором свойств – хэшем ключа, значением ключа, значением элемента словаря.

**Обратите внимание.**

Неизменяемые объекты с равными значениями в Python получают одинаковые хэши.

\* Объекты с разными значениями могут получить одинаковые хэши (что является хеш-коллизией).

```
1. a = {}  
2. a["key"] = 1  
3. a[1] = 0  
4. a[1.0] = "sphere"
```

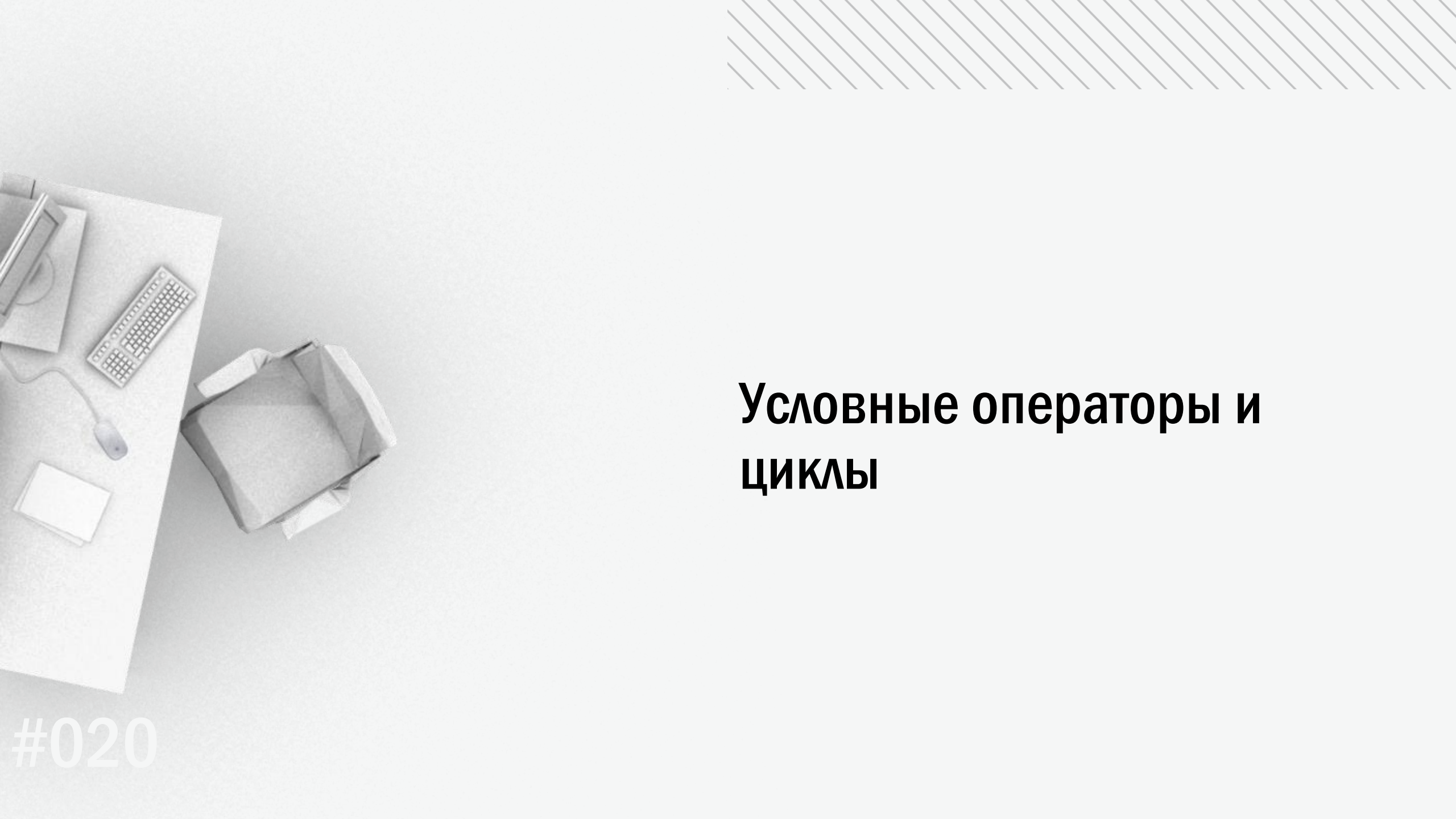
# Python: [Множества] set

Множества (set) являются неупорядоченной коллекцией из уникальных элементов.

Внутри CPython множества реализованы как хэш-таблицы, в которых есть только ключи без значений, за счёт чего и обеспечивается уникальность элементов.

Множества являются изменяемым итерируемым объектом.

```
1. a = {1, 2, 3}
```



# Условные операторы и циклы

#020

# Python: Условные инструкции

Блоки условных инструкций выделяются отступами или табуляцией.

```
1. a = 1.1
2. if 2 > a > 1:
3.     res = True
4. else:
5.     res = False
```

Краткая форма записи:

```
1. a = 1.1
2. res = True if 2 > a > 1 else False
```

# Python: Циклы

В python есть 2 типа циклов:  
while и for.

Пример while:

```
1. while условие:  
2.     блок инструкций
```

Пример for:

```
1. for i in range(10):  
2.     блок инструкций
```



Блок инструкций может содержать инструкции управления циклом: `continue` (перейти к следующей итерации), `break` (прервать выполнение цикла).

Вне основного блока инструкций цикла можно разместить `else`, блок инструкций под которым будет выполняться всегда.

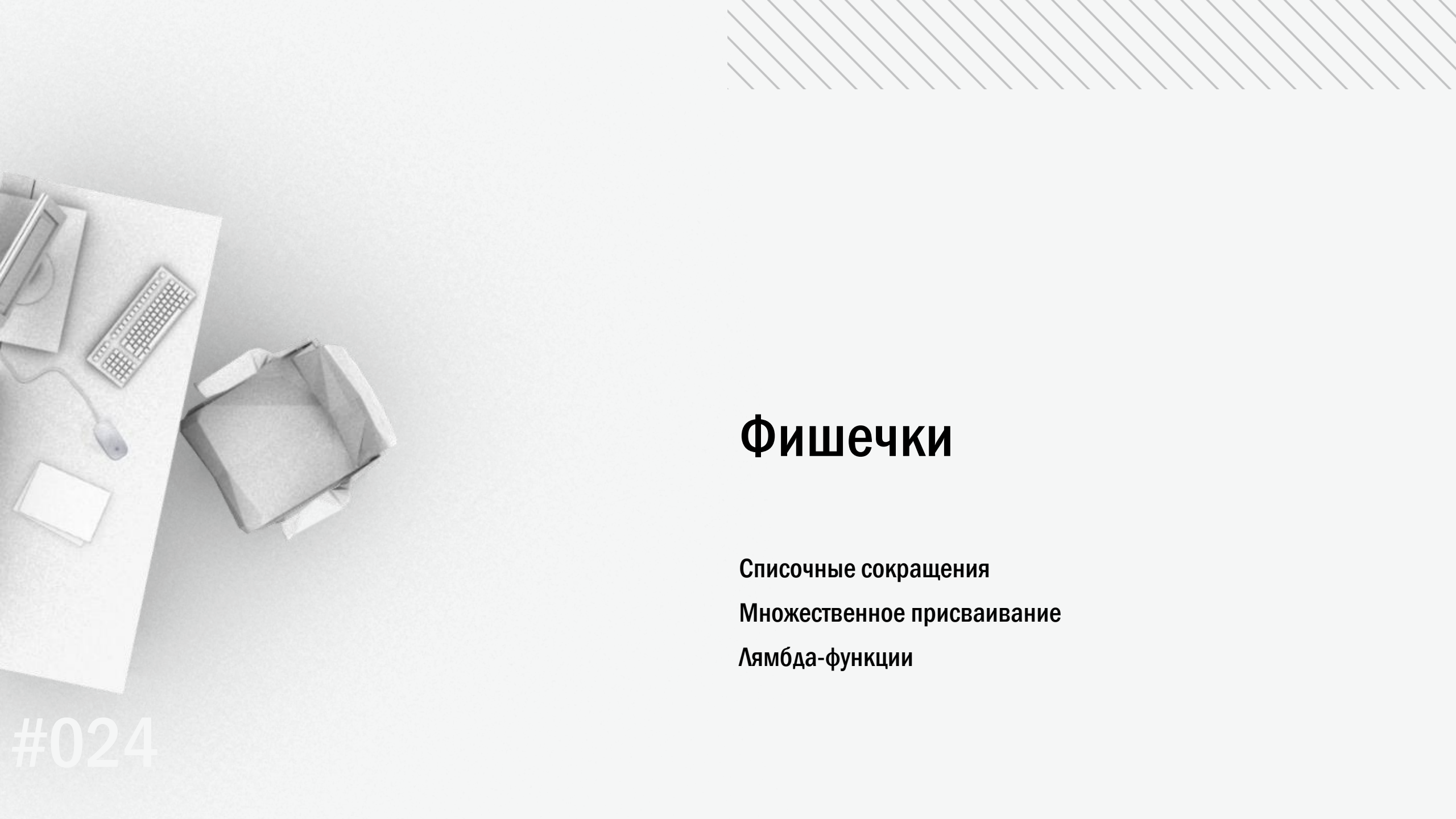
# Python: Функции

Для объявления функции служит ключевое слово «def».

Аргументы функции задаются в скобках после названия функции. Можно задавать необязательные аргументы, присваивая им значение по умолчанию.

Ключевое слово «lambda» служит для объявления анонимных функций.

```
1. def example(x1, x2=0):  
2.     return x1**2+x2
```



# Фишечки

Списочные сокращения

Множественное присваивание

Лямбда-функции

#024



# Python: List comprehensions (списочные сокращения)

Списковые включения (list comprehensions) являются встроенным в Python механизмом генерации списков.

Реализуется следующим образом:

```
1. squares = [x**2 for x in range(10)]
```

Можно заменить на выражение с функцией map.

```
1. squares = list(map(lambda x: x**2, range(10)))
```

Разница в областях видимости и поэтому map безопаснее.

# Python: Множественное присваивание

В Python можно делать множественное присваивание, распаковку строк, списков, кортежей, словарей.

```
1. a, b = 0, 1
```

=

```
1. a, b = [0, 1]
```

В случае если переменных меньше, чем элементов объекта, который вы хотите распаковать:

```
1. a, b, *c = 0, 1, 2, 3
```

```
1. a, *b, c = 0, 1, 2, 3
```

# Python: Лямбда-функции

Для объявления функции служит ключевое слово «def».

Аргументы функции задаются в скобках после названия функции. Можно задавать необязательные аргументы, присваивая им значение по умолчанию.

Ключевое слово «lambda» служит для объявления анонимных функций.

```
1. def example(x1, x2=0):  
2.     return x1**2+x2
```

=

```
1. example = lambda x1, x2=0: x1**2+x2
```

---

# Python: Импорты

Пример импорта модуля

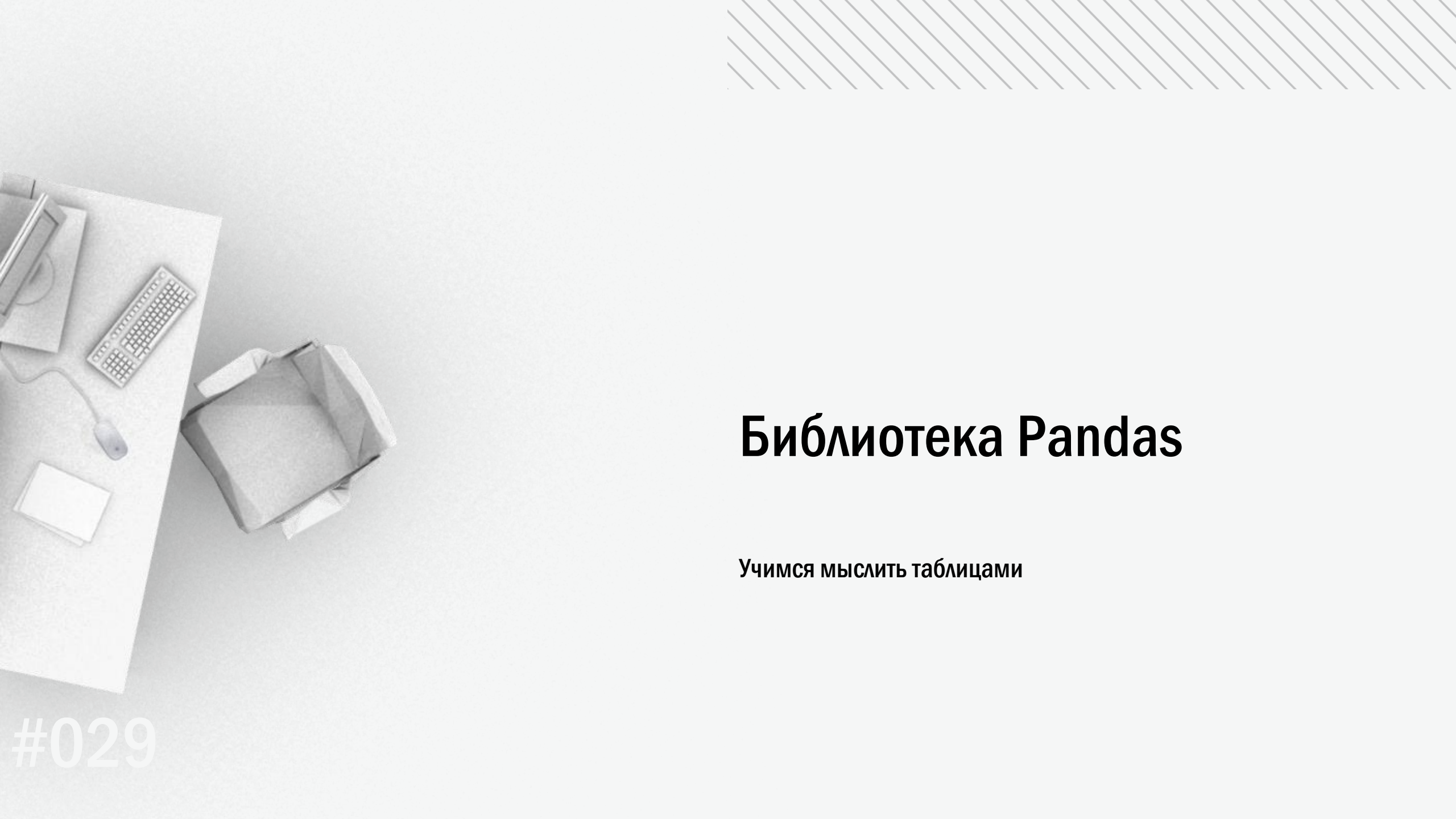
```
1. import this
```

Импорт дочерних модулей.

```
1. from numpy import random
```

=

```
1. import numpy.random
```



# Библиотека Pandas

Учимся мыслить таблицами

#029

# Pandas – работаем с таблицами данных

```
1 data = pd.read_csv('winequality-red.csv')
2 data.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

# Pandas – работаем с таблицами данных

```
1 data.shape
```

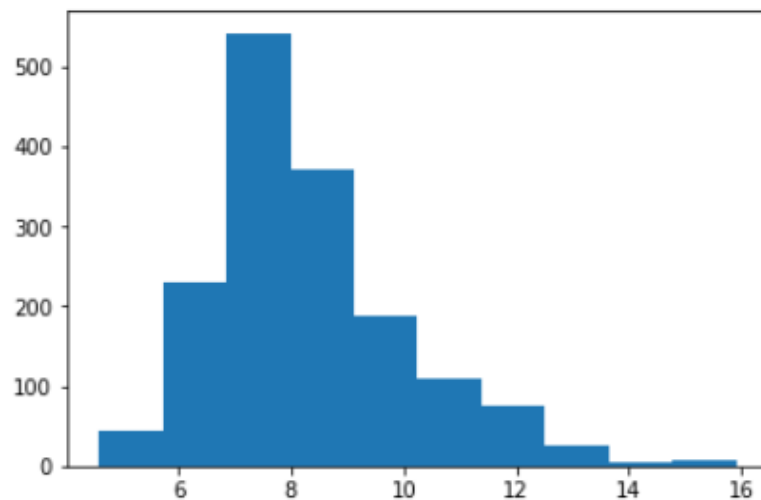
```
(1599, 12)
```

```
1 data['fixed acidity'].mean()
```

```
8.319637273295838
```

```
1 plt.hist(data['fixed acidity'])
```

```
(array([ 45., 229., 542., 371., 188., 110., 76., 26., 5., 7.]),  
array([ 4.6 ,  5.73,  6.86,  7.99,  9.12, 10.25, 11.38, 12.51, 13.64,  
        14.77, 15.9 ]),  
<a list of 10 Patch objects>)
```



---

- **Домашнее задание**

Ищем данные для проекта на Kaggle  
Читаем данные в Python



**СПАСИБО  
ЗА ВНИМАНИЕ**

