Type()

Id()



Abs()

Pow()

Import math

Math.sqrt()

Math.ceil()

Math.floor()

A,b = map(int, intput().split())

Print(a, b, c, sep = “|”)

Print(a, end = “ ”)

F-строки:

s1, s2 = map(str.strip, input().split())

print(f"Word 1: {s1} | Word 2: {s2}")

Bool()

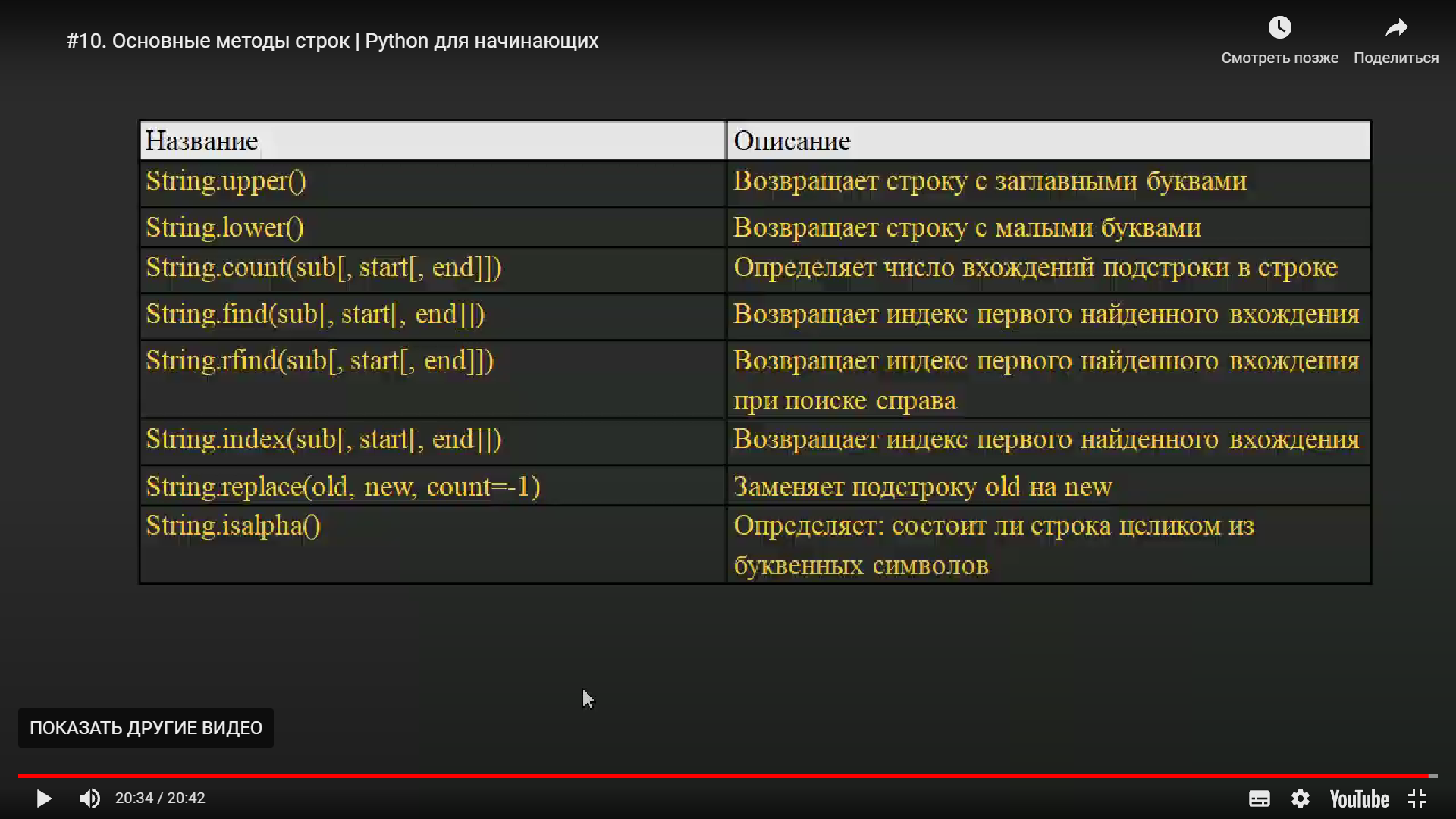
Print(s1 + “ ” + s2)

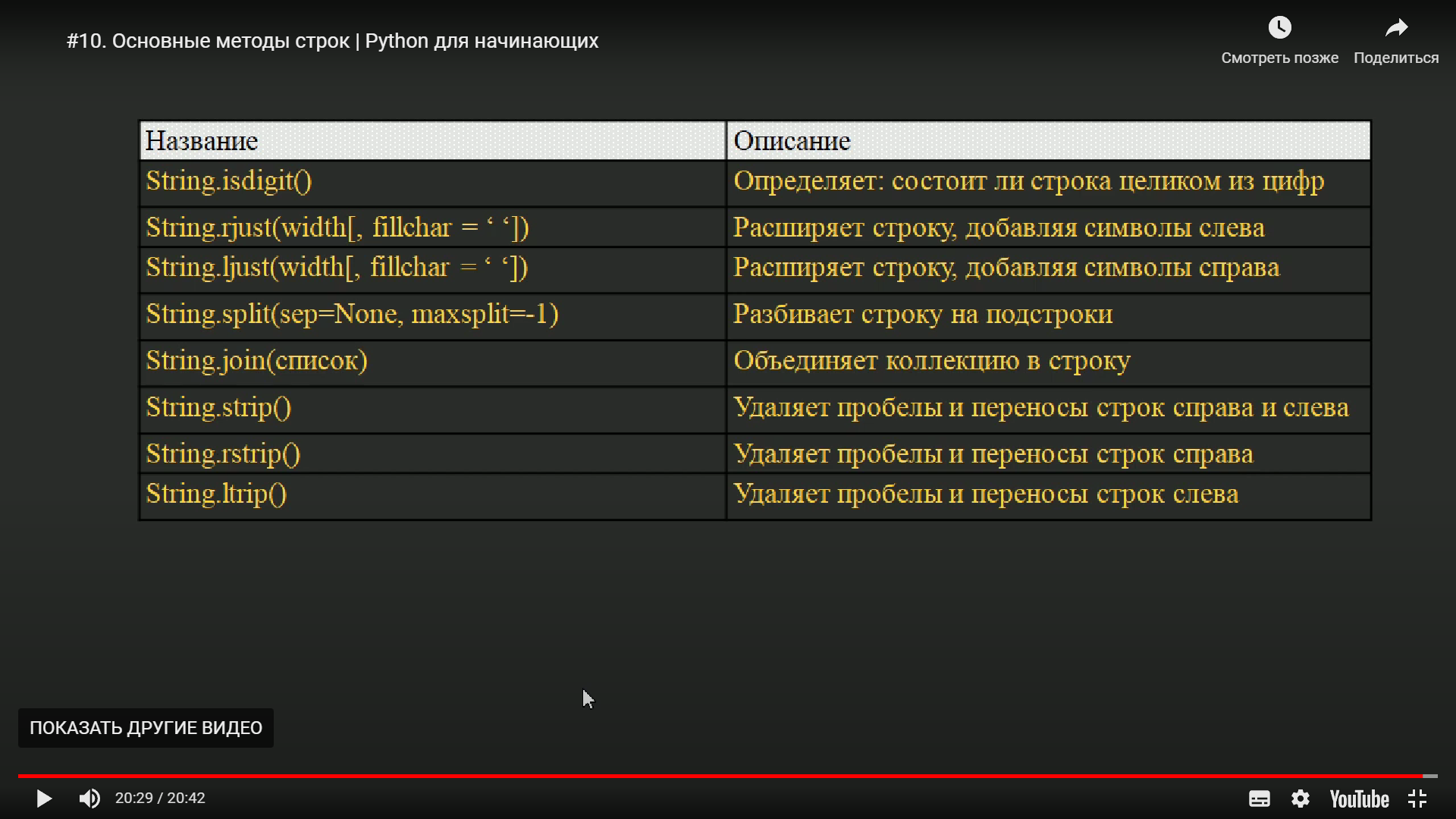
Коммент из нескольких строк - в тройных одинарных или тройных двойных кавычках

Len()

Ord()

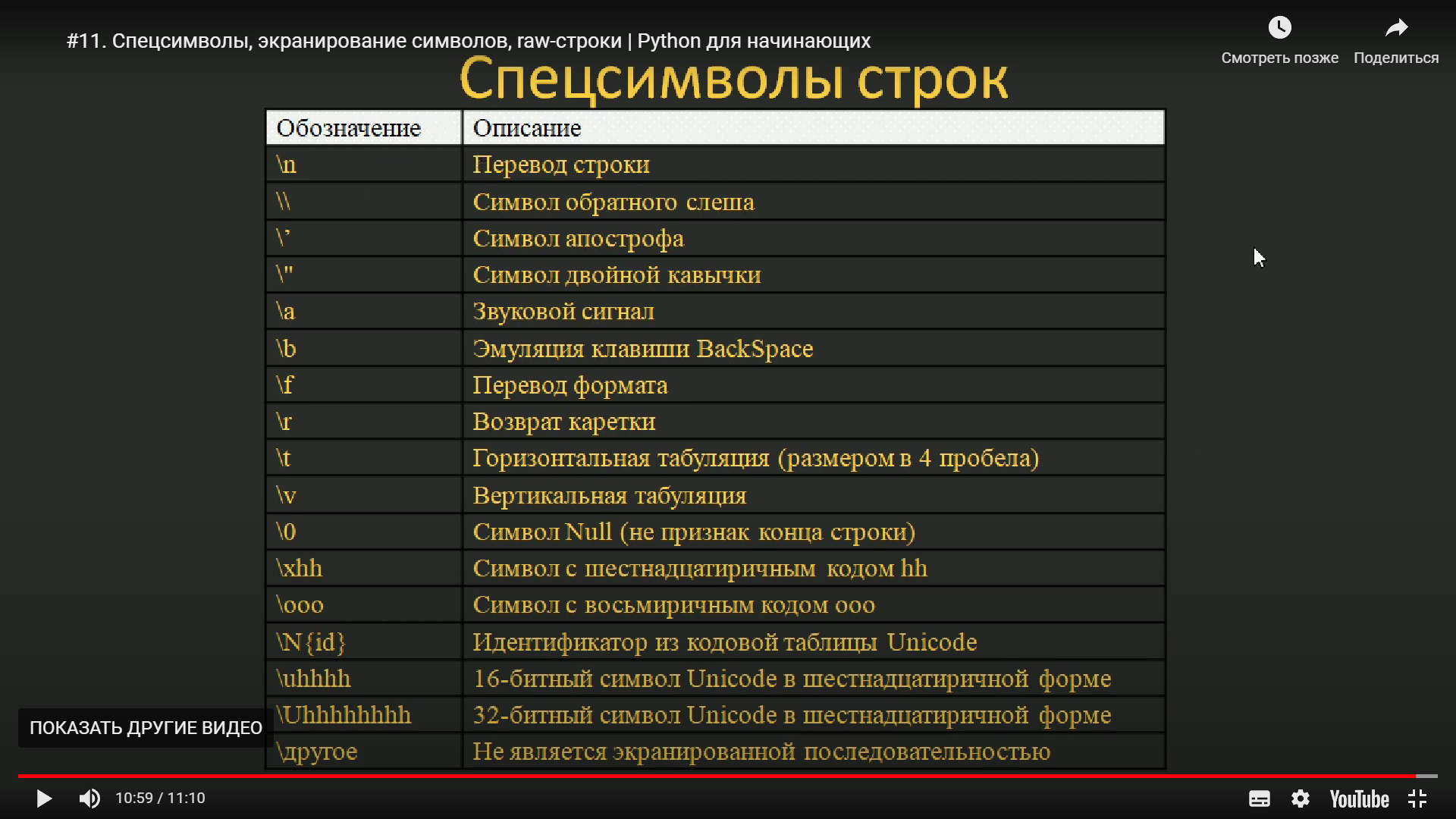
Str[1:5:-1]





“ ”.join(lst) – преобразует список в строку, между элементами в данном случае пробел

Если наоборот - list(“python”)



Path = r”D:usrer\stepik\trip”

Print(path) – выведет тоже самую строку, игнорируя спецсимволы

“меня зовут {0}, мне {1} лет”.format(name, age)

“меня зовут {fio}, мне {number} лет”.format(fio = name, number = age)

Список – тот же массив, только не нужно указывать тип и в нем могут быть объекты любого типа

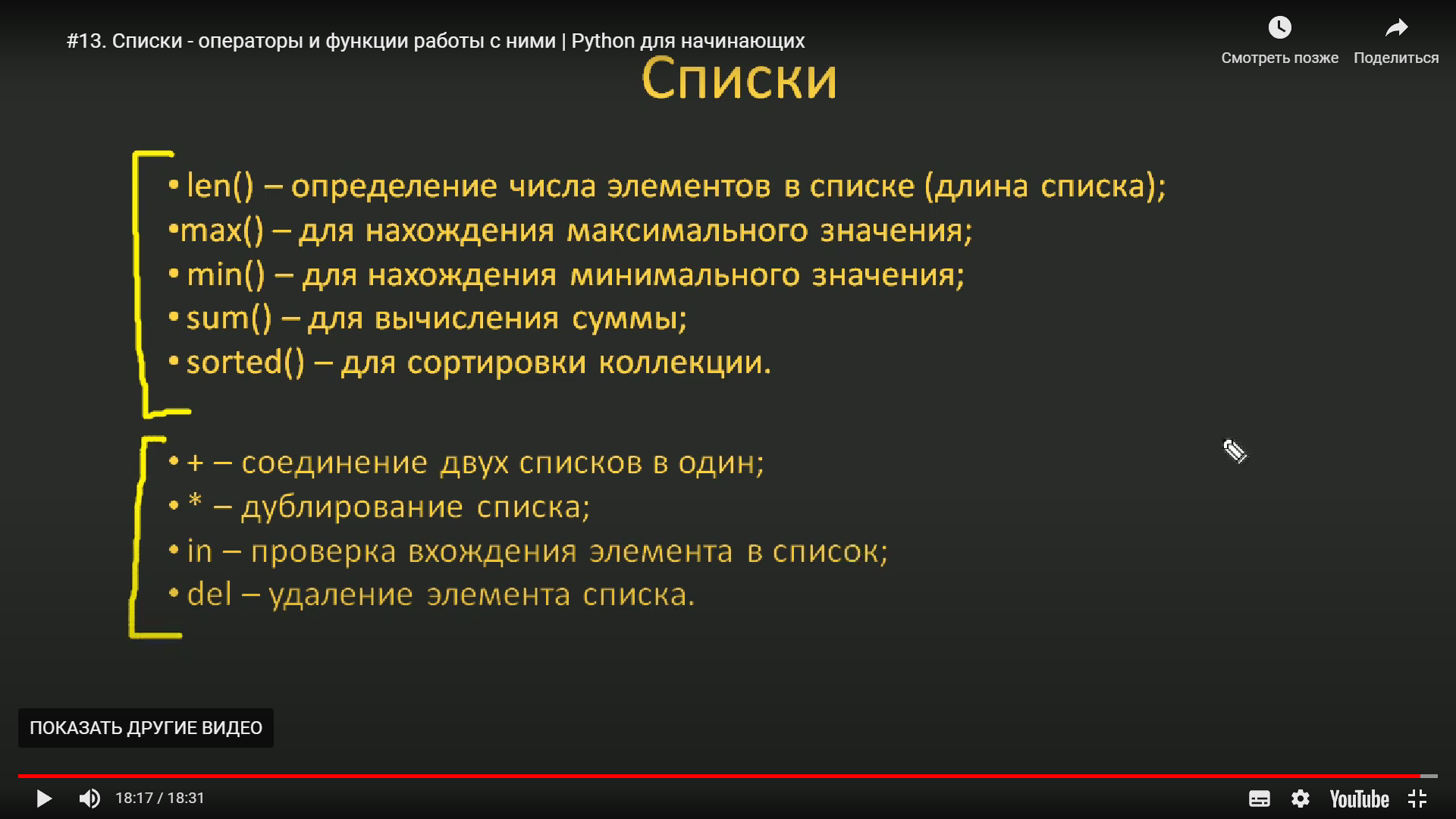
a[True, “sdds”, 1, 6]

a[] или a = list() – пустой список

a = list(“python”) – [p, y, t, h, o, n]

a[1] – True

a[-1] – 6



lst = sorted(lst, reverse = True) – в обратном порядке

del lst[2] – удаляет прямо в списке безвозвратно

список – тоже самое, что строка

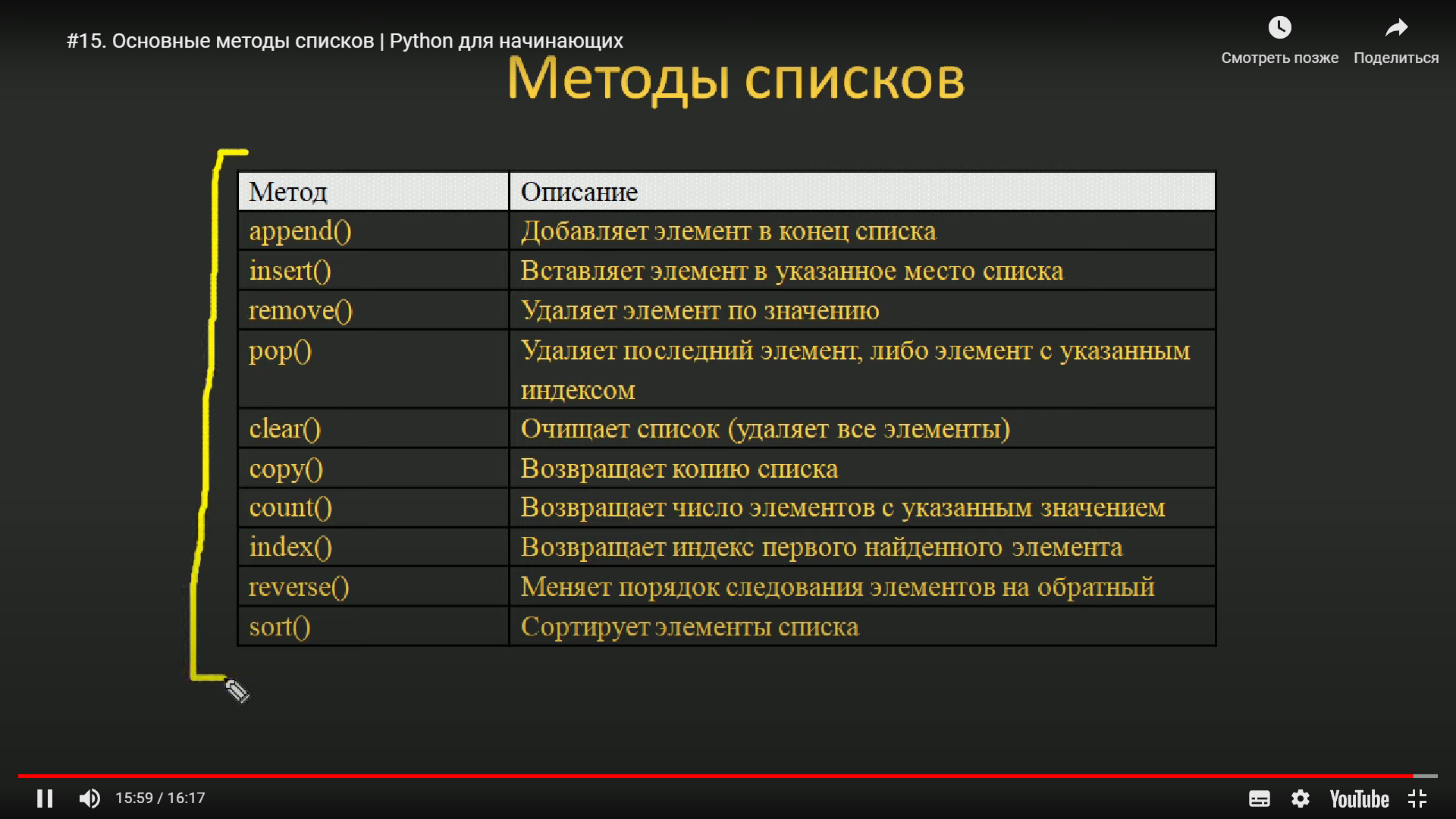
lst[1:5:-1]

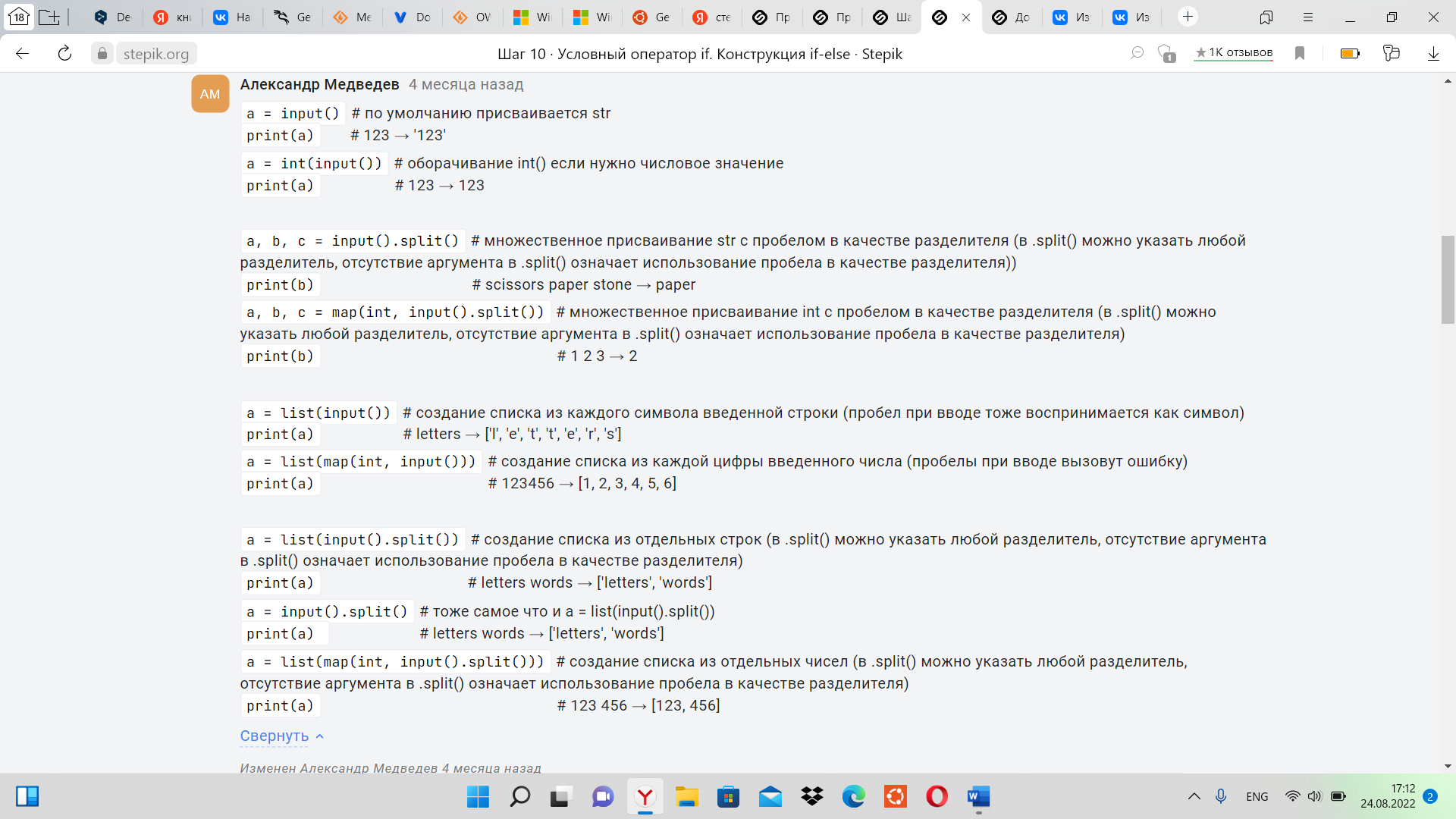
lst[1:5:2] = [0,0,0]

lst = lst1 – только тогда будет ссылаться на один и тот же объект с одним адресом id

marks[2:4] = 10, 20

можно сравнивать различные списки – работает как строки





If x < 0:

Print()

Else:

Print()

If x < 0:

Print()

Elif 0 <= x <= 9:

Print()

Elif 10<=x<=99

Print()

Тернарный условный оператор

A + 2 if a > c else c - 3 – можно вставлять его везде

Break and continue

p = [0] \* 10

i = 0

while p.count(1) != 5:

if p.count(1) == 5:

break

i = int(input())

if p[i] == 0:

p[i] = 1

else:

continue

print(\*p)

while … :

else:

цикл for и range()

lst = []

for i in range(11):

lst.append(i)

print(\*lst) – выведет числа от 0 до 10 включительно

for i in [1,2,3,4,5]:

“ “.join(words) – быстро составить из списка строку с разд пробелы

D = “10 + 12”

print(eval(d)) – вычисляет строку

функция enumerate – передает 2 параметра индекс и значение

lst = list(map(int, input().split(" ")))

for i, d in enumerate(lst):

lst[i] = d\*\*2

print(\*lst)

**Итератор и итерируемые объекты. Функции iter и next**

lst = list(input().split())

it = iter(lst)

print(next(it))

print(next(it))

a = [“четное” if x % 2 == 0 else “нечетное”

for x in d

if x > 0

]

5.9 Вложенные циклы и вложенные генераторы списков

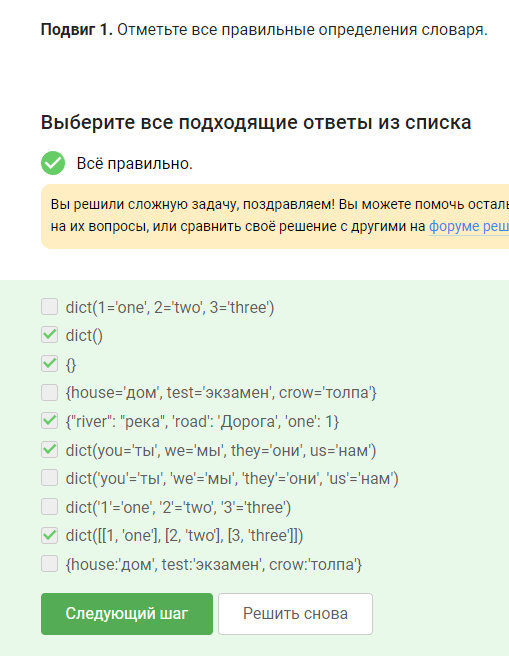
Пример – транспонирование матрицы

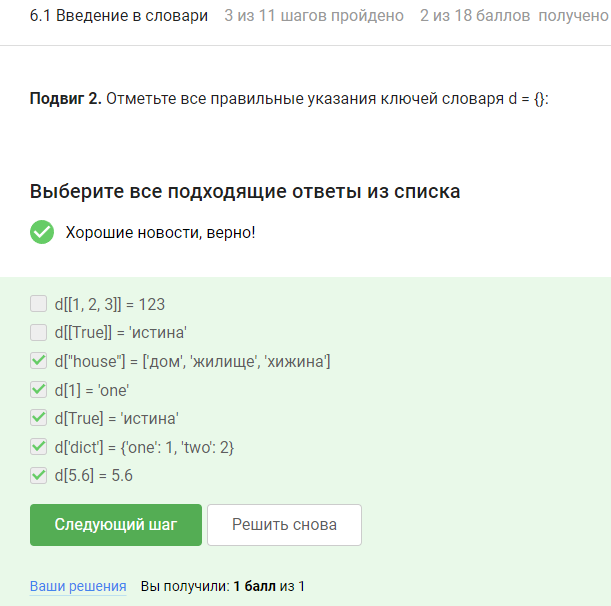
A = [[1,2,3,4],[5,6,7,8],[9,10,11,12]] A = [[row[i] for row in A] for i in range(len(A[0]))] print(A)

Словари

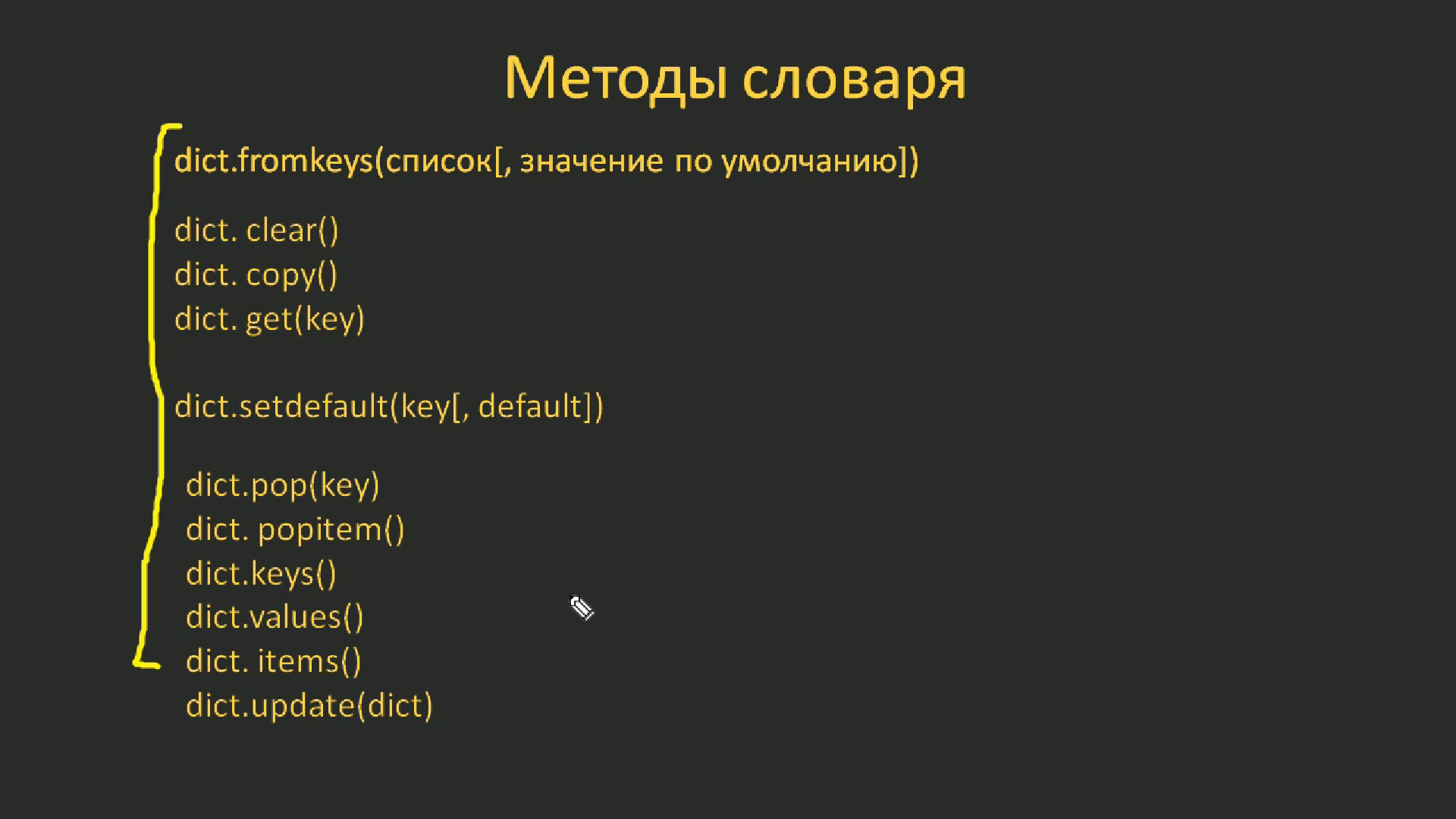
{"river": "река", 'road': 'Дорога', 'one': 1}

dict(you='ты', we='мы', they='они', us='нам')



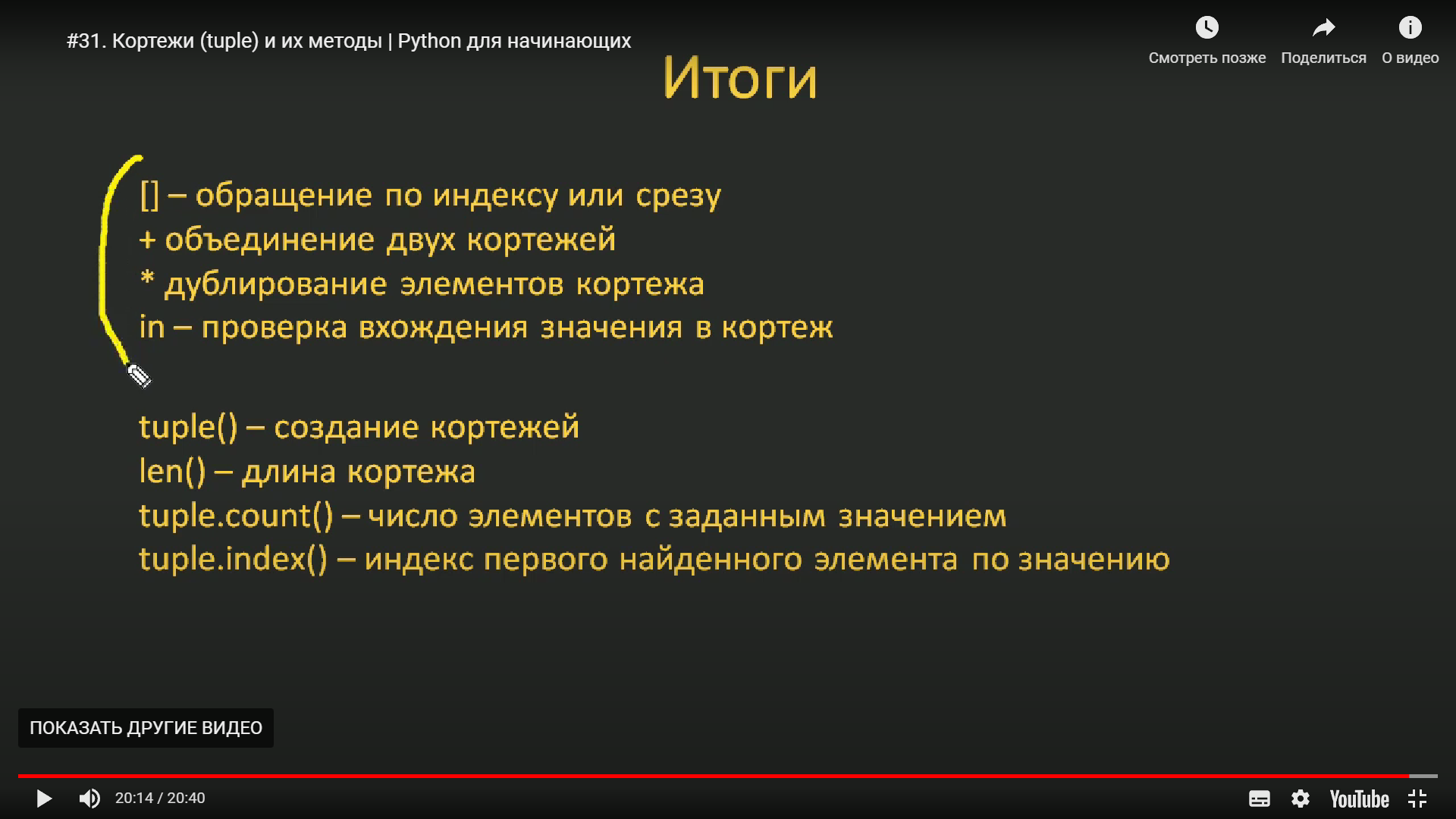


Методы словарей



1. Создает словарь из списка, значения, что были в списке становятся ключами, если не прописывать значение по умолчанию, то значения во всех ключах становятся none, если прописать то во всех которое прописано
2. Очищает полностью словарь
3. Копирует словарь на другой адрес
4. Возращает значение по ключу, если ключ отсутствует то возвращает none, d.get(key, False) – в этом случае если отсутсвует то возвращает false
5. Также возвращает значение по ключу, но если ключ отсутствует то создает этот ключ в словаре с заданым значением, если значения нет то со значением none и также возвращает это значение
6. Удаляет ключ с значением по ключу, возвращает значение удаленного ключа
7. Удаляет любой ключ наугад
8. Возвращает все ключи словаря
9. Возвращает все значения словаря
10. Возвращает и ключи и значения словаря
11. Добавляет словарь в скобках в словарь перед точкой

Кортеж – упорядоченный, неизменяемый



A = (1,2,3)

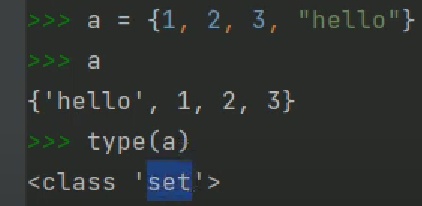
B = 1,

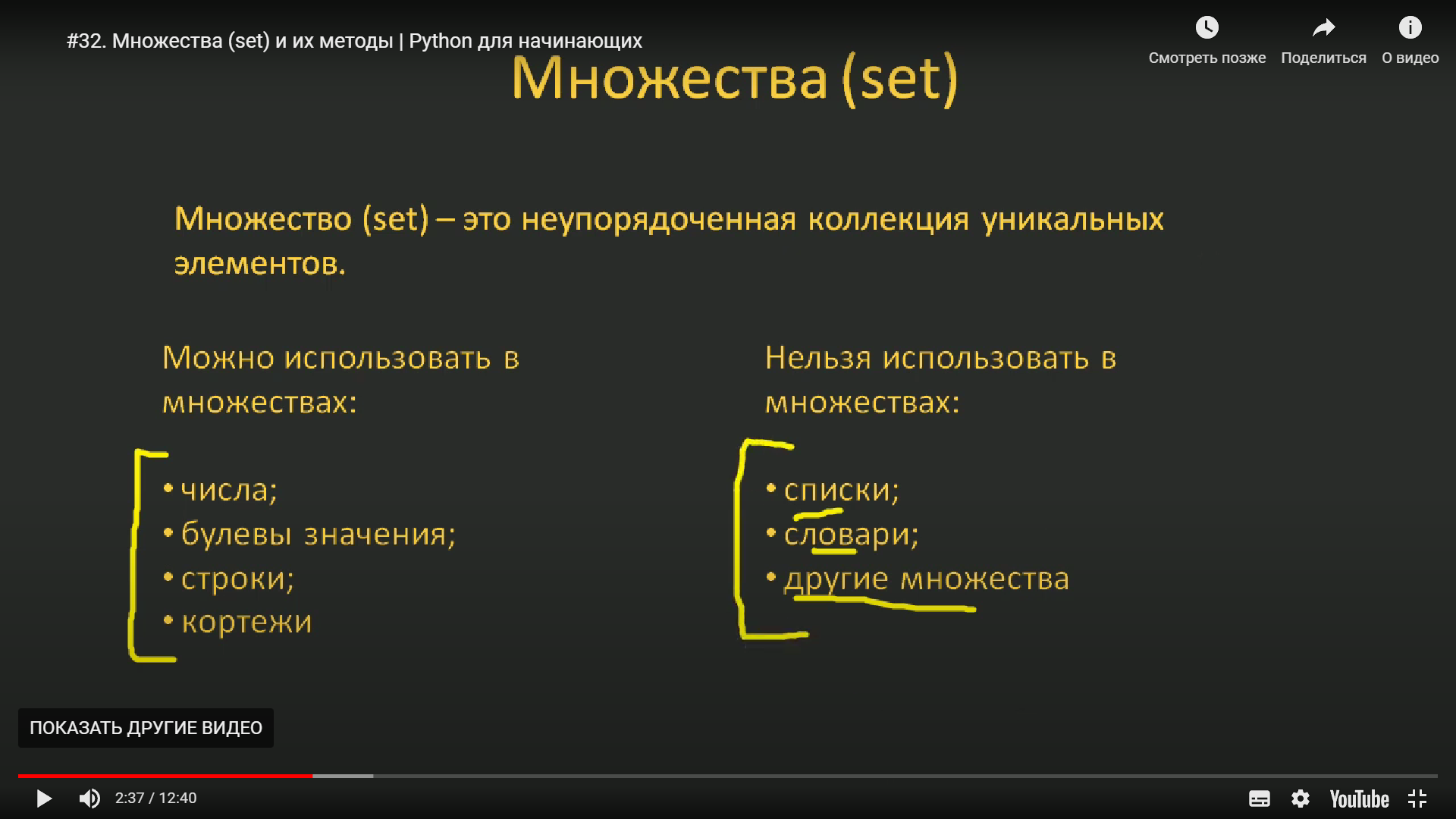
A[0:2] – выведет (1,2) – поддеживает срезы как и списки

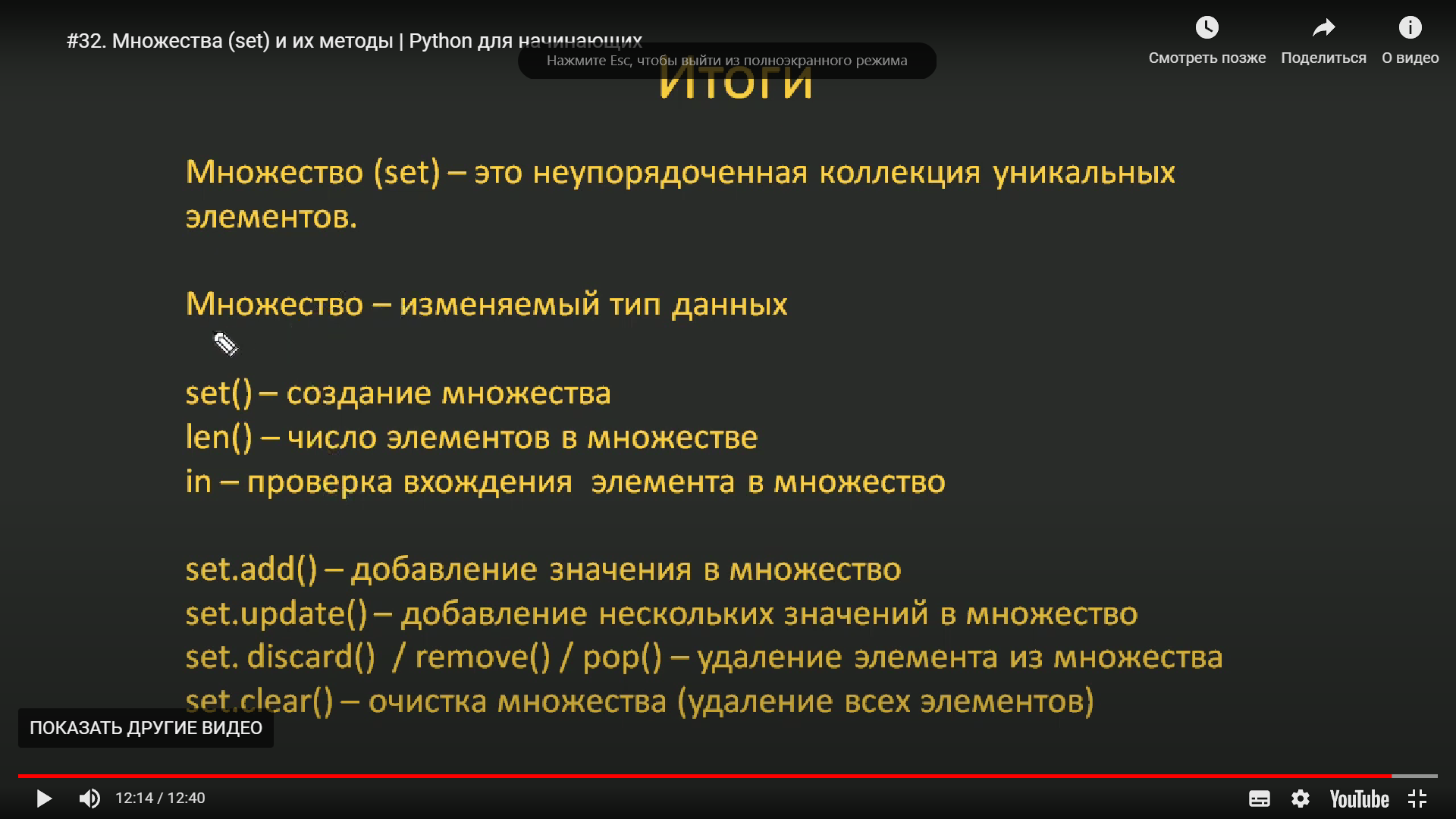
A = tuple(“hhhff”) - превращает любой элемент в кортеж

С = (True, [1,2,3]) c[1].append(5) - C станет (True, [1,2,3,5]) – то есть можно менять изменяемые данные внутри кортежа

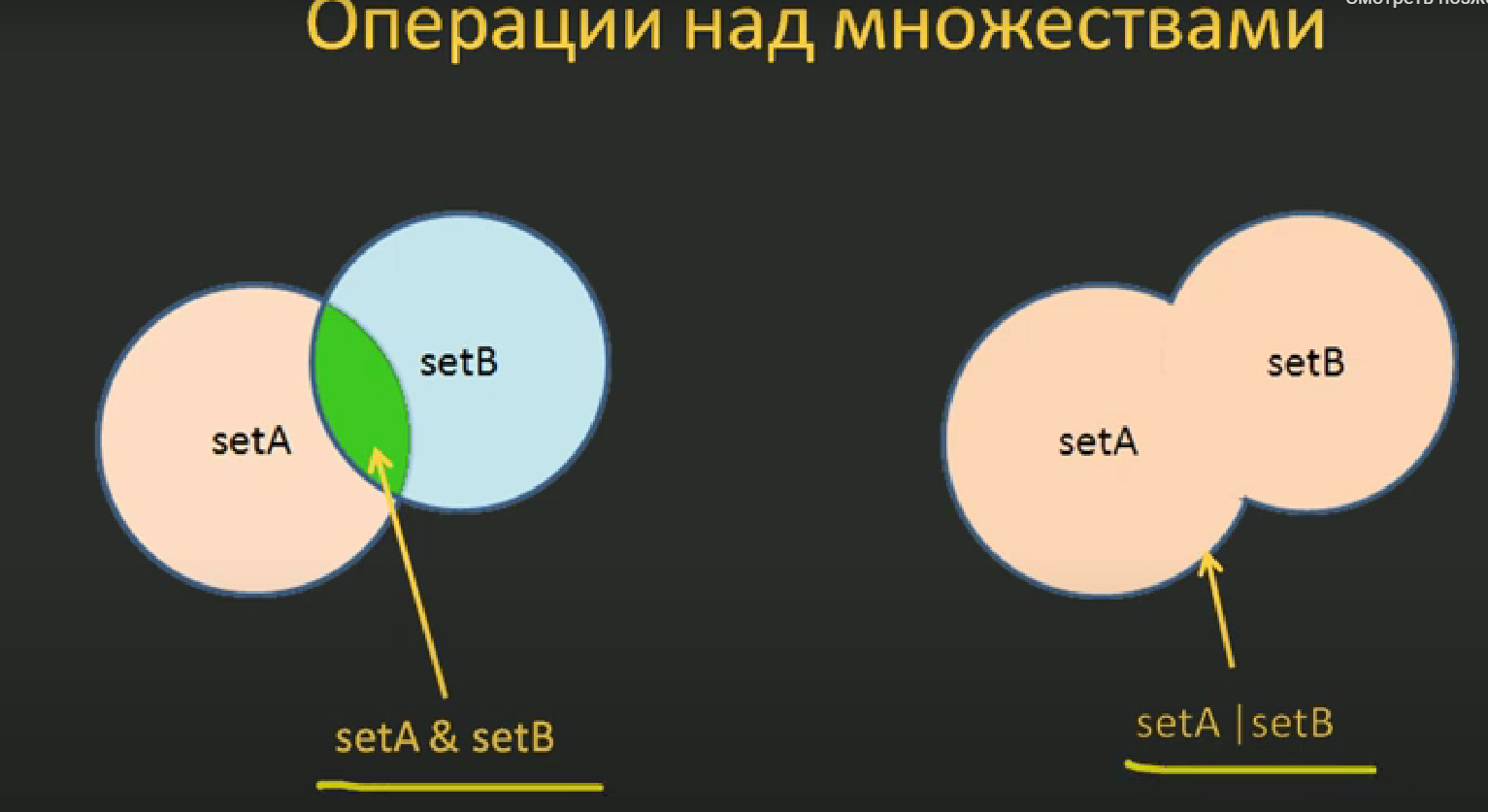
Множество – неупорядоченная коллекция уникальных элементов







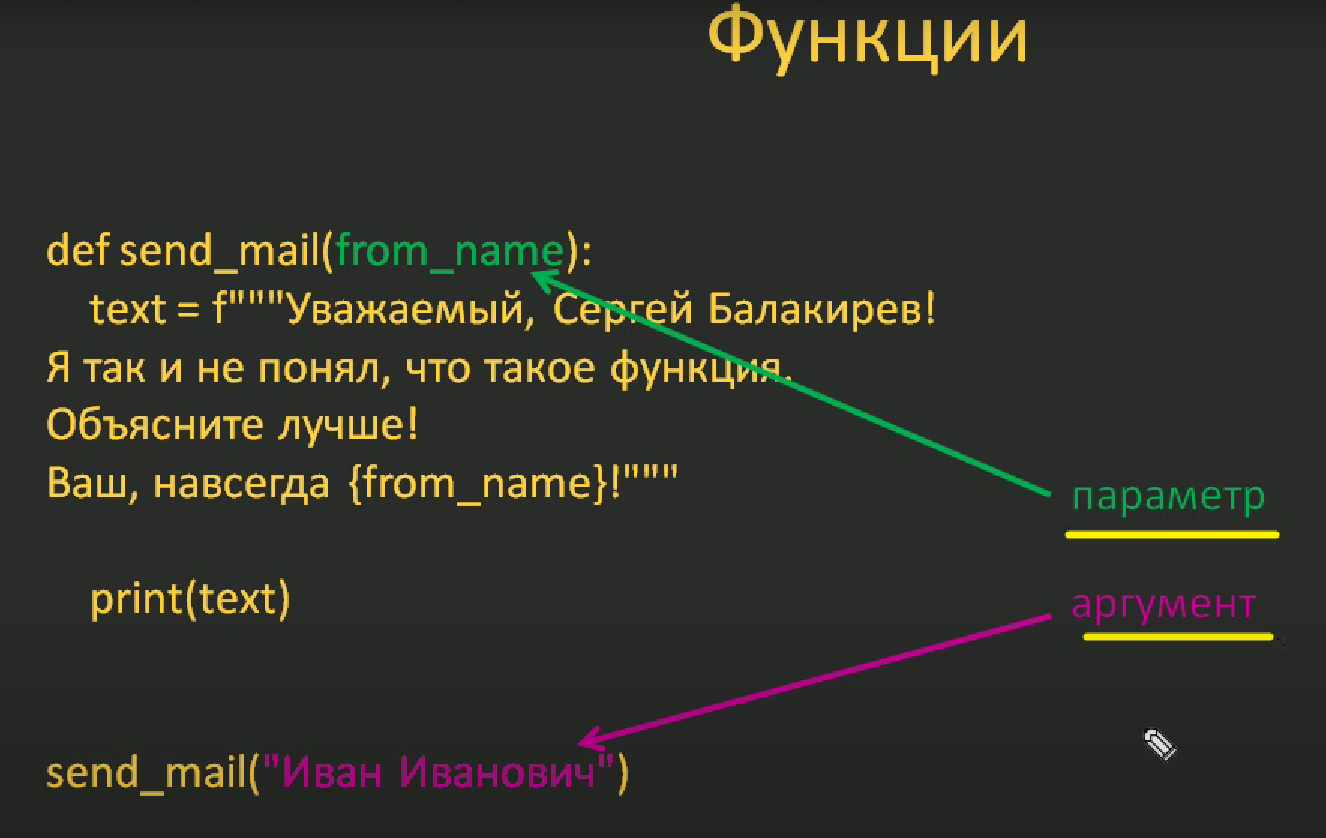
Операции над множествами

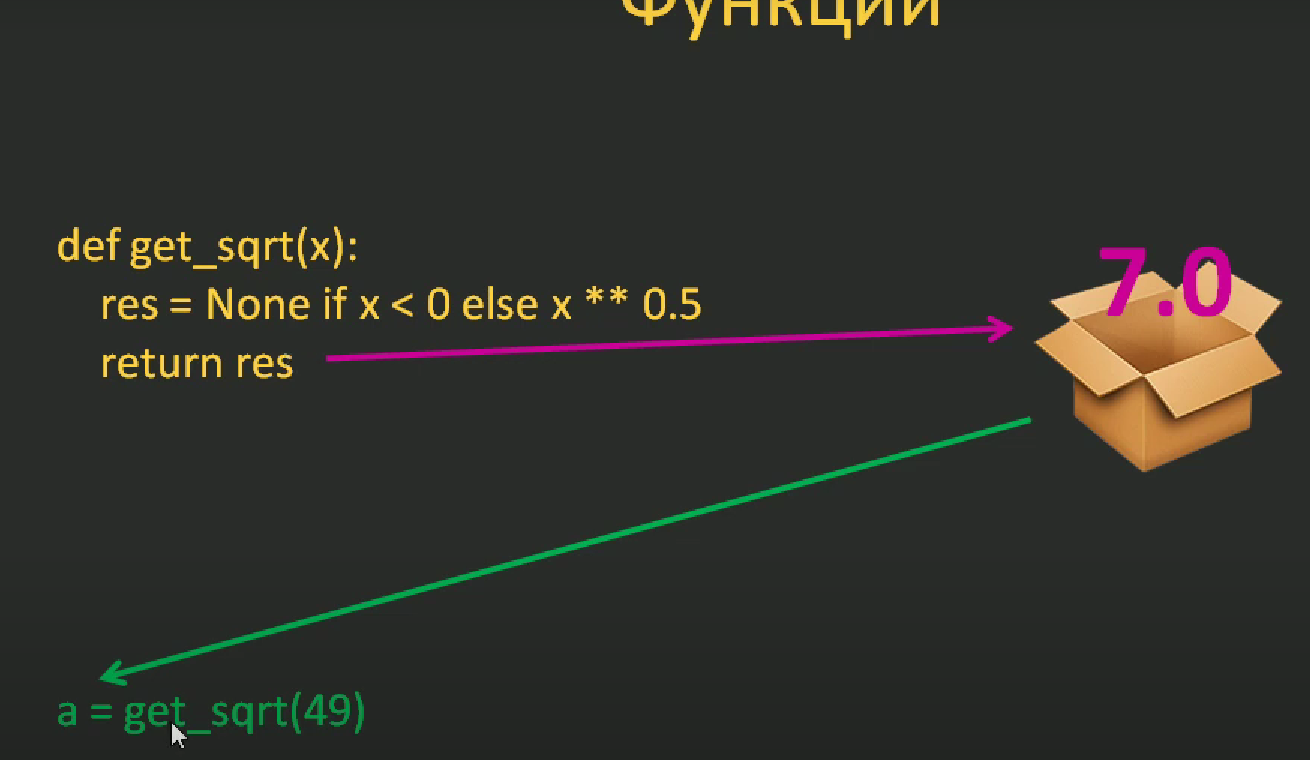


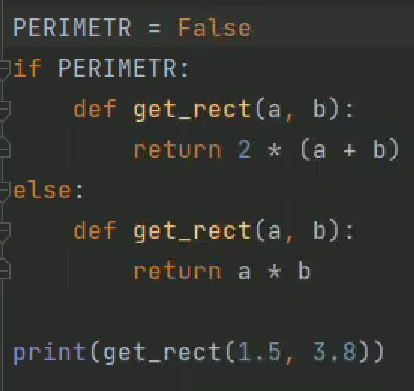


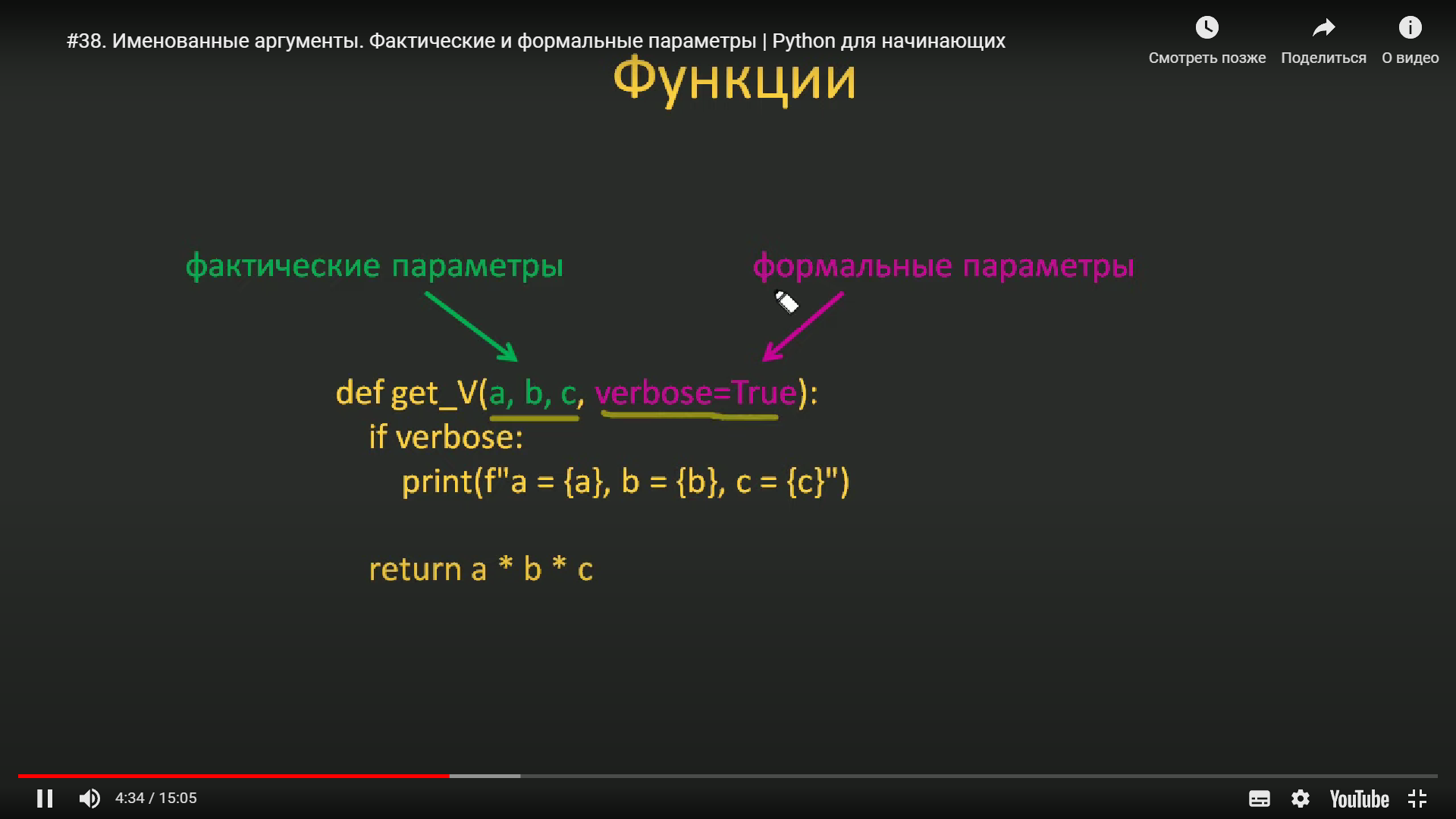
A = {1,2,3} b = {1,3,4,2} a < b возвратит true тк а является подмн-вом b

Функции



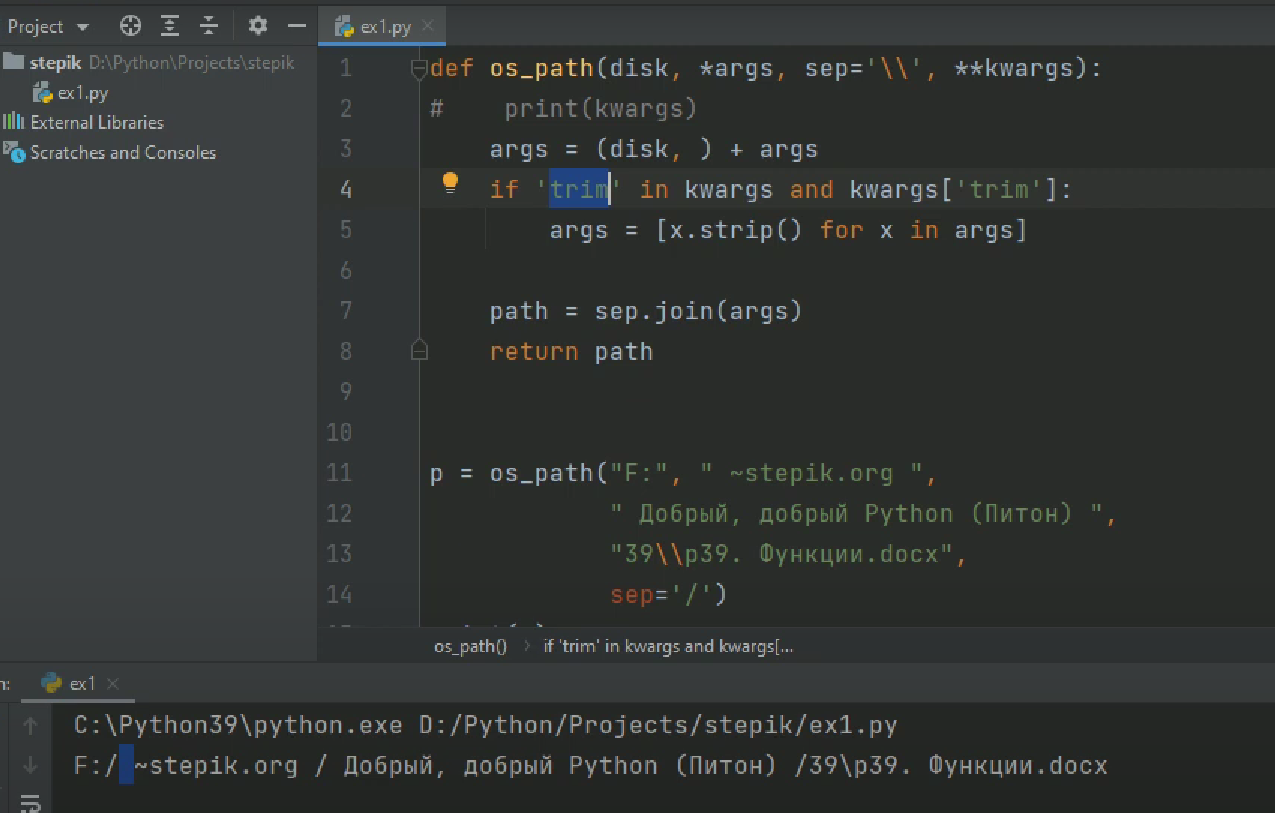




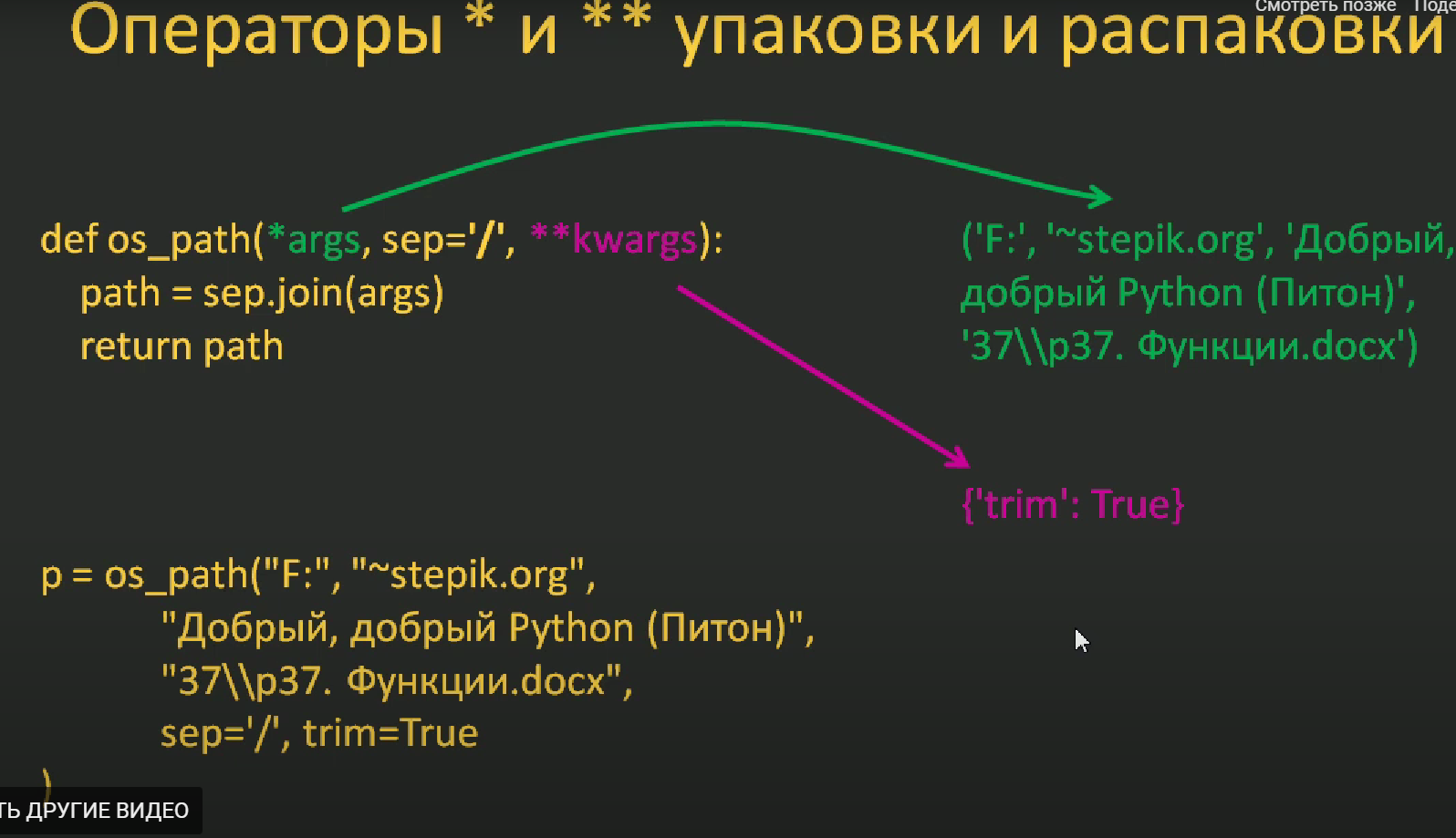


# Функции с произвольным числом параметров

\*args \*\*kwargs



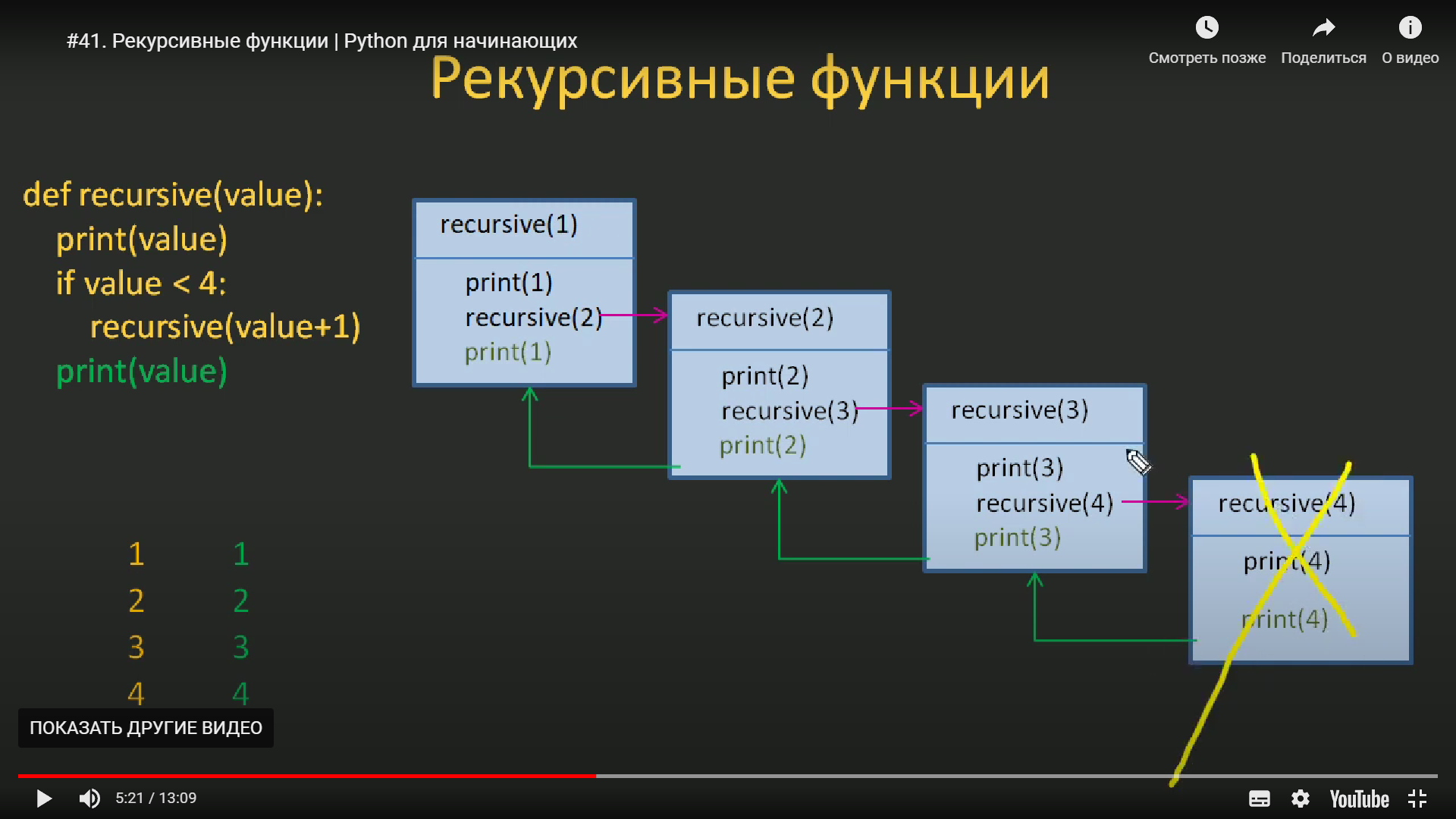
Операторы упаковки и распаковки



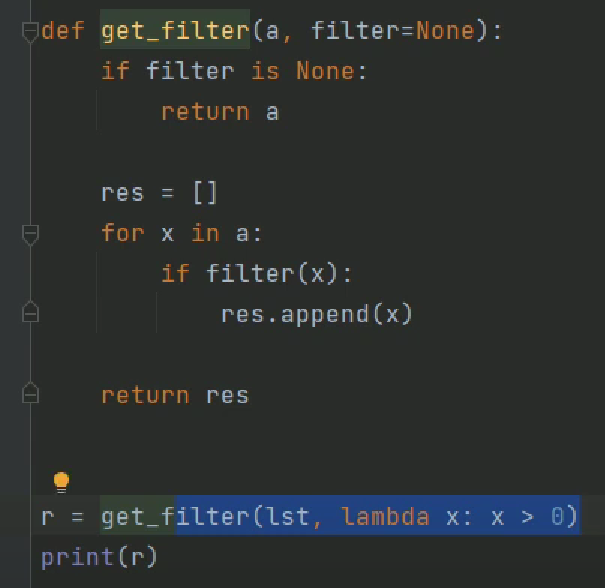
A, b, \*c = 1,2,3,4,5,6,7,8

{\*range(-5, 5), \*(True, False), \*str} – {-5, -4,…, True, False, p, y, t, h, o, n}

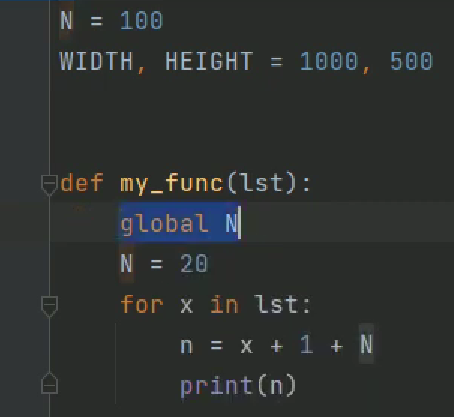
Рекурсии

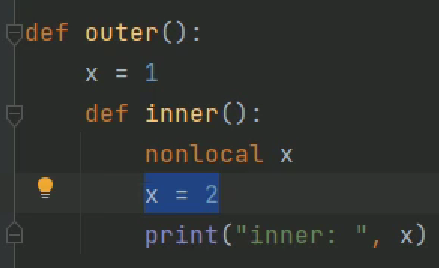


Анонимные лямбда функции



# Области видимости. Ключевые слова global и nonlocal



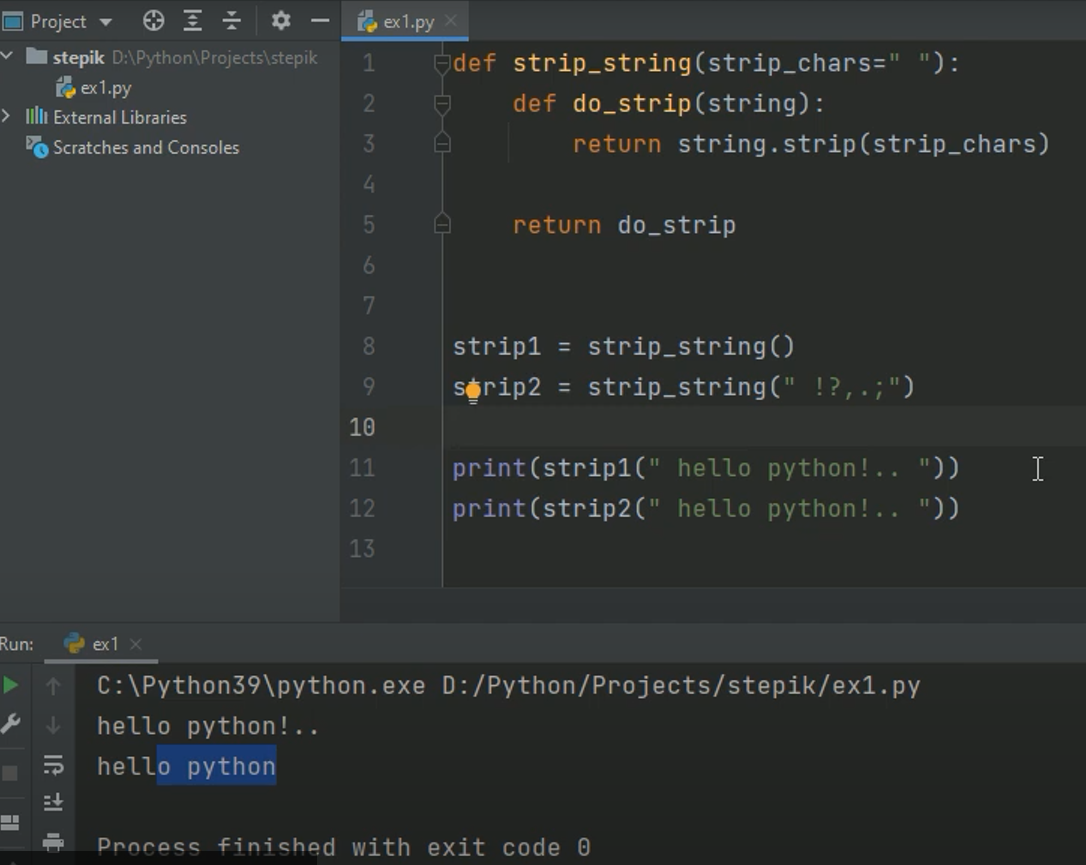


Пишем global – если нужно изменить глобальную переменную

Nonlocal – если изменить локальную на уровне выше

# Замыкания в Python. Вложенные функции

С помощью одной функции можем делать несколько вычислений или преобразований



# Декораторы функций – благодаря им можно сократить количество строк кода. Декораторы это как бы надбавки к функциям, которые мы можем удобно использовать для разных ф-ций. То есть использовать одну функцию декоратор для разных функций.

# 

# 

# Передача аргументов декораторам

# Чтобы передавать дополнительные переменные в декоратор – например значение дельта х для производной

# 

# 

# Чтобы вывести название функции – специальна япеременная \_\_name\_\_

# 

# Sin\_df.\_\_name\_\_

# Чтобы вывести комментарий функции – переменная - \_\_doc\_\_

# 

# Чтобы декоратор не менял имя и комментарий функции

# 

# Или так

# 

# 

# 8й раздел

# Импорт стандартных модулей. Команды import и from

# Import math

# - если нужны только несколько функций

# Можно импортировать свои созданные модули

# 

# Pip list – команда в терминале показывающая установленные пакеты

# Pip install -r requirements.txt – устанавливает пакеты с файла

# Чтобы сформировать данный файл:

# Pip freeze > requirements.txt

# 

# Импорт пакета – также как и модуля, в пакете то есть директории содержится спец файл \_\_init\_\_.py – в нем прописываются импорты всех нужных модулей или нужных функций из модулей

# 

# 

# Метод file.read(4) – считыват первые 4 символа

# Метод file.seek(0) – обнуляет местоположение считывания символов, иначе метод read так и продолжит считывать содержимое файла

# Метод file.tell() – возвращает текущее местоположение считывателя, но не по символам, а по байтам

# 

# Метод file.readline() – считывает одну строку файла вместе символом переноса строки, если явно не указать конец файла

# Прочитать все строчки:

# 

# 

# Метод file.readlines() – формирует список из строчек файла

# Также после откытия файла нужно обязательно его закрыть и тем самым освободить место в программе от содержимого файла – метод file.close() – в конце

# 

# 

# След менеджер контекста with аналогичен прошлому коду + сам закрывает файл

# 

# Запись файла – атрибут – r – если файл не существует – то он создается. Если в файле содержатся данные то он перезаписыватется

# 

# Чтобы не перезаписывать файл а добавлять в него новые значения – нужно поменять атрибут на - a

# 

# Чтобы одновременно иметь возможность добавлять данные в файл и читать его – атрибут а+

# 

# Запись списка в двоичный файл с помощью библ-ки pickle

# 

# И наоборот чтение данных из бинарного списка, обрати внимание на атрибут добавляется буква b

# 

# Многократная запись в двоичный файл

# 

# Многократное чтение данных из файла в бинарном режиме доступа

# 

# - метод записи в файл нескольких слов или строк, можно даже им записать список

# 9й раздел

# Выражения-генераторы

# Создание генераторов в круглых скобках – создается ссылка на генератор – генератор не формирует значения сразу а видимо хранит только диапазон и формулу формирования значений– чтобы посмотреть значения формируемые генератором – нужно list(a) – создать из него список – но сгенерировать значения можно только один раз

# 

# Функция-генератор. Оператор yield

# В данным случае а в цикле обращвется к функции и занаво берет значение в а

# 

# Программа для поиска индекса слов

# 

# Функция map – возвращает сформированный генератор, в ней пишем функцию и итерируемый тип, эта фунция преобразовывает значения согласно этой функции

# 

# 

# Функция filter

# 

# 

# Функция zip

# 

# 

# 

# 9.6 Сортировка с помощью sort и sorted

# a.sort() – меняет значение а, ничего не возвращает

# sorted(a) – не меняет значения в а, возвращает отсортированный итер объект

# a.sort(reverse = True)

# sorted(a, reverse = True)

# Аргумент key для сортировки по ключу

# 

# 

# Функция isinstance для проверки типов данных

# 

# 

# 

# Функции all и any

# All – если все переменные True то возращает true

# Any – если хотя бы одна переменная true то возвращает true

# 

# Расширенное представление чисел

# 

# 

# Битовые операции И, ИЛИ, НЕ, XOR

# 

# 

# 

# Отнимается дополнительно еденица

# 

# 

# 

# 

# 

# 

# При побитовом сдвиге вправо число делится на 2

# Влева умножается на 2 в степени сдвига

# 