# Основа основ

**pip install pytube**

**from pytube import Youtube**

# Скачати відео

**yt = YouTube('http://youtube.com/watch?v=2lAe1cqCOXo')** - Теперь у нас есть YouTubeобъект с именем yt (Силка яка хочете, тут вона просто до приклада)

## Отримати назву відео

**yt.title** – щоб отримати назву відео

yt.thumbnail\_url – щоб отримати URL-адрес миниатюры

## Розширені можливості при створенні обєкта Youtube

yt = YouTube(

'http://youtube.com/watch?v=2lAe1cqCOXo',

on\_progress\_callback=progress\_func,

on\_complete\_callback=complete\_func,

proxies=my\_proxies,

use\_oauth=False,

allow\_oauth\_cache=True

)

Функция **on\_progress\_callback** будет запускаться всякий раз, когда фрагмент загружается из видео, и вызывается с тремя **аргументами: поток, фрагмент данных и байты**, оставшиеся в видео. Это можно использовать, например, для отображения индикатора выполнения.

Функция **on\_complete\_callback** запускается после полной загрузки видео и вызывается с двумя **аргументами: потоком и путем к файлу**. Это можно использовать, например, для обработки видео после загрузки, например обрезки его длины.

Флаги **use\_oauth** и **allow\_oauth\_cache** позволяют вам разрешить pytube взаимодействовать с YouTube с использованием вашей учетной записи и могут использоваться для обхода возрастных ограничений или доступа к частным видео и плейлистам. Если для параметра **allow\_oauth\_cache** установлено значение **True**, вам будет предложено сделать это только один раз, после чего pytube будет кэшировать токены, необходимые для действия от вашего имени. В противном случае вам будет снова предложено выполнить каждое действие, требующее аутентификации.

# Работа с потоками и StreamQuery

Как упоминалось ранее, **прогрессивные потоки** содержат видео и аудио в одном файле, но обычно не обеспечивают мультимедиа высочайшего качества; в то же время **адаптивные потоки** разделяют видео и аудио дорожки, но могут обеспечить гораздо более высокое качество.

Начните с выполнения следующей команды, чтобы **получить список всех потоков**:

**yt.streams**

### Вивести кожний поток через \n:

for el in yt.streams:

print(el)

## Фильтрация потоков

**.filter()** – щоб фільтрувати по потокам

### Фильтрация по потоковому методу

**yt.streams.filter(progressive=True)**  - фильтровать только прогрессивные потоки

**yt.streams.filter(adaptive=True)** – фільтрувати только потоки DASH (также называемые «адаптивными»)

### Фильтрация потоков только аудіо

**yt.streams.filter(only\_audio=True)** - Чтобы запросить потоки, содержащие только аудиодорожку

### Фильтрация потоков MP4

**yt.streams.filter(file\_extension='mp4')**  - Чтобы запросить только потоки в формате MP4

## Загрузка потоков

**stream = yt.streams.get\_by\_itag(22) –** Обрати обєкт по itag

**stream.download()** – Скачати обєкт

# Дорожки субтитров/титров

Начнем с переключения на видео, которое их содержит

**yt.captions**

Теперь давайте проверим английские подписи:

**caption = yt.captions.get\_by\_language\_code('en')**

Отлично, теперь посмотрим, как YouTube их форматирует:

**caption.xml\_captions**

давайте конвертируем их в формат srt:

**caption.generate\_srt\_captions()**

**print(caption.generate\_srt\_captions())**

# Использование плейлистов

## Создание списка воспроизведения

**from** **pytube** **import** Playlist

Теперь давайте создадим объект списка воспроизведения. Вы можете сделать это, инициализировав объект URL-адресом списка воспроизведения:

**>>>** p = Playlist('https://www.youtube.com/playlist?list=PLS1QulWo1RIaJECMeUT4LFwJ-ghgoSH6n')

Или вы можете создать его из ссылки на видео в плейлисте:

**>>>** p = Playlist('https://www.youtube.com/watch?v=41qgdwd3zAg&list=PLS1QulWo1RIaJECMeUT4LFwJ-ghgoSH6n')

Теперь у нас есть **Playlist**объект, pс которым мы можем поработать.

## Взаимодействие с плейлистом

По сути, объект Playlist — это просто контейнер для объектов YouTube.

### Скачати плейліст

Если, например, мы хотим загрузить все видео в плейлисте, мы должны сделать следующее:

**>>>** print(f'Downloading: *{*p.title*}*')

Downloading: Python Tutorial for Beginers (For Absolute Beginners)

**>>> for** video **in** p.videos:

**>>>**  video.streams.first().download()

### Інше

Или, если нас интересуют только URL-адреса видео, мы можем просмотреть и их:

**>>> for** url **in** p.video\_urls[:3]:

**>>>**  print(url)

['https://www.youtube.com/watch?v=41qgdwd3zAg',

'https://www.youtube.com/watch?v=Lbs7vmx3YwU',

'https://www.youtube.com/watch?v=YtX-Rmoea0M']

# Использование каналов

## Создание канала

**from** **pytube** **import** Channel

Теперь давайте создадим объект канала. Вы можете сделать это, инициализировав объект URL-адресом канала:

**>>>** c = Channel('https://www.youtube.com/c/ProgrammingKnowledge')

Или вы можете создать его из ссылки на видео-страницу канала:

**>>>** c = Channel('https://www.youtube.com/c/ProgrammingKnowledge/videos')

Теперь у нас есть **Channel**объект, cс которым мы можем поработать.

## Взаимодействие с каналом

По сути, объект Channel — это просто контейнер для объектов YouTube.

### Скачати відео з канала

Если, например, мы хотим загрузить все видео, созданные на канале, мы должны сделать следующее:

**>>>** print(f'Downloading videos by: *{*c.channel\_name*}*')

Downloading videos by: ProgrammingKnowledge

**>>> for** video **in** c.videos:

**>>>**  video.streams.first().download()

### Інше

Или, если нас интересуют только URL-адреса видео, мы можем просмотреть и их:

**>>> for** url **in** c.video\_urls[:3]:

**>>>**  print(url)

['https://www.youtube.com/watch?v=tMqMU1U2MCU',

'https://www.youtube.com/watch?v=YBfInrtWq8Y',

'https://www.youtube.com/watch?v=EP9WrMw6Gzg']

# Использование функции поиска

## Основне

**from** **pytube** **import** Search

s = Search('YouTube Rewind')

**>>>** len(s.results)

17

**>>>** s.results

[\

<pytube.\_\_main\_\_.YouTube object: videoId=YbJOTdZBX1g>, \

<pytube.\_\_main\_\_.YouTube object: videoId=PKtnafFtfEo>, \

...\

]

Из-за возможности бесконечного потока результатов и во избежание случайного входа пользователя в **бесконечный цикл** запроса дополнительных результатов атрибут **.results**

будет запрашивать только первый набор результатов поиска. Дополнительные результаты можно явно запросить с помощью **.get\_next\_results()** метода, который добавит любые дополнительные результаты к **.results** атрибуту

## Дополнительный функционал

В дополнение к основным функциям поиска, которые возвращают объекты YouTube, поиск также имеет соответствующие предложения автозаполнения. Доступ к ним можно получить следующим образом:

**>>>** s.completion\_suggestions

[\

'can this video get 1 million dislikes', \

'youtube rewind 2020 musical', \

...\

]

# Интерфейс командной строки (CLI)

Pytube также поставляется с крошечным интерфейсом командной строки для взаимодействия с видео и плейлистами.

Чтобы загрузить прогрессивный поток самого высокого разрешения:

$ pytube <https://www.youtube.com/watch?v=2lAe1cqCOXo>

Чтобы просмотреть доступные потоки:

$ pytube https://www.youtube.com/watch?v=2lAe1cqCOXo –list

Чтобы скачать конкретный поток, используйте itag

$ pytube https://www.youtube.com/watch?v=2lAe1cqCOXo --itag=22

Чтобы получить список всех субтитров (кодов субтитров)

$ pytube https://www.youtube.com/watch?v=2lAe1cqCOXo --list-captions

Чтобы загрузить конкретный субтитр (код подписи) — в данном случае английские субтитры (в формате srt) — используйте:

$ pytube https://www.youtube.com/watch?v=2lAe1cqCOXo -c en

Также можно просто загрузить аудиопоток (по умолчанию AAC/mp4):

$ pytube https://www.youtube.com/watch?v=2lAe1cqCOXo -a

Чтобы просмотреть все параметры командной строки, просто введите

$ pytube --help

Наконец, если вы отправляете отчет об ошибке, в интерфейсе командной строки есть переключатель --build-playback-report, который объединяет состояние, позволяя другим легко воспроизвести вашу проблему.

# Обработка исключений

 Вообще говоря, если видео по какой-либо причине недоступно, это можно отловить с помощью общего исключения **VideoUnavailable**. Это можно использовать, например, для пропуска частных видео в плейлисте, видео с региональными ограничениями и т. д.

Давайте посмотрим, как может выглядеть ваш код, если вам нужно выполнить обработку исключений:

**>>> from** **pytube** **import** Playlist, YouTube

**>>> from** **pytube.exceptions** **import** VideoUnavailable

**>>>** playlist\_url = 'https://youtube.com/playlist?list=special\_playlist\_id'

**>>>** p = Playlist(playlist\_url)

**>>> for** url **in** p.video\_urls:

**...**  **try**:

**...**  yt = YouTube(url)

**...**  **except** VideoUnavailable:

**...**  print(f'Video *{*url*}* is unavaialable, skipping.')

**...**  **else**:

**...**  print(f'Downloading video: *{*url*}*')

**...**  yt.streams.first().download()

Это автоматически пропустит видео, которые не удалось загрузить из-за ограничений библиотеки pytube.

# API

## Объект YouTube

**.author** - Узнайте автора видео

**.bypass\_age\_gate() -** Попытайтесь обновить vid\_info, минуя возрастные ограничения

**.caption\_tracks** - Получите список Caption.

**.check\_availability( ) -** Проверьте, доступно ли видео.

Вызывает различные исключения в зависимости от того, почему видео недоступно, в противном случае ничего не делает.

**.description -** Получите описание видео

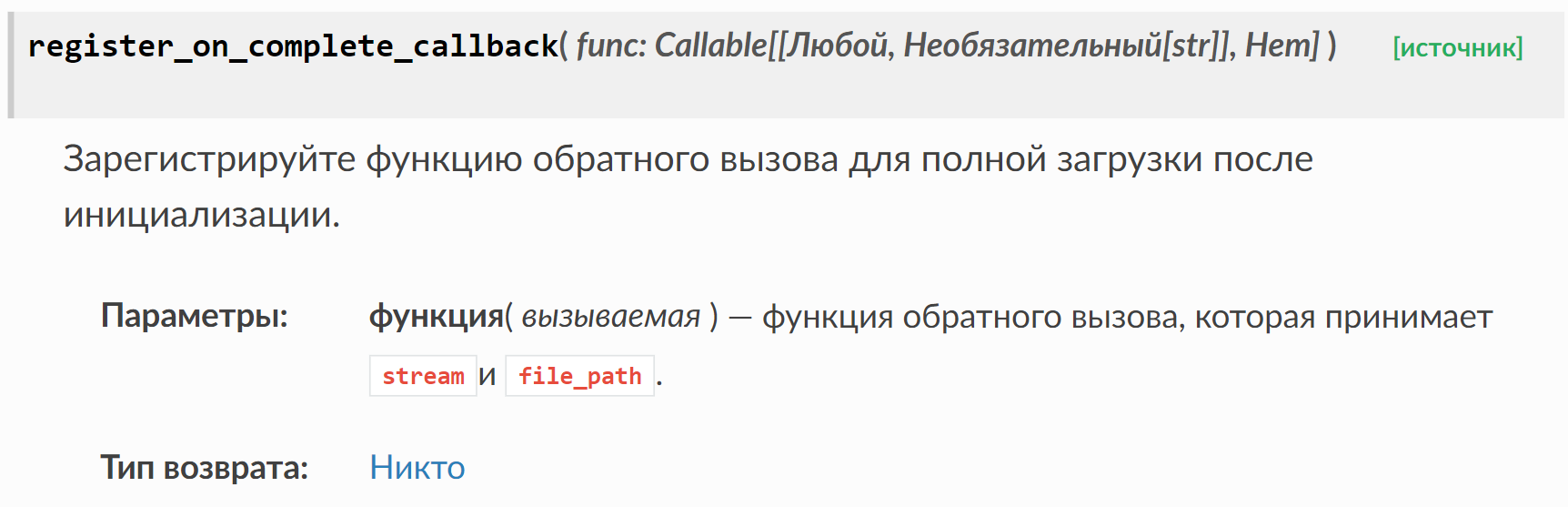
**.Keywords** - Получите ключевые слова для видео.

**.length** - Получите длину видео в секундах

**.publish\_date** - Получите дату публикации.

**.rating** - Получите средний рейтинг видео.

**register\_on\_complete\_callback( func: Callable[[Любой, Необязательный[str]], Нет] )**



**.streams** - Интерфейс для запроса как адаптивных (DASH), так и прогрессивных потоков.

**.thumbnail\_url** - Получите URL-адрес уменьшенного изображения.

**.title** - Получите название видео.

**.views** - Получите количество просмотров видео.

## Объект списка воспроизведения/Playlist

**.length** - Извлеките количество видео в плейлисте.

**.owner** - Извлеките владельца плейлиста.

**.owner\_id** - Извлеките Channel\_id владельца плейлиста.

**.owner\_url** - Создайте URL-адрес канала владельца плейлиста.

**.title** - Извлечь название плейлиста

**.video\_urls** - Полные ссылки на все видео в плейлисте

**.videos** - Создает объекты YouTube для видео в этом плейлисте.

**.views** - Извлечь количество просмотров плейлиста.

## Объект канала

**.channel\_name** - Получите название канала YouTube.

## Объект потока

**pytube.Stream(поток: Dict[KT, VT], моностат: pytube.monostate.Monostate) -** Контейнер для данных манифеста потока.

**.default\_filename** - Создайте имя файла на основе названия видео.

**download( выходной\_путь: Необязательный[строка] = Нет , имя файла: Необязательный[строка] = Нет , префикс имени\_файла: Необязательный[строка] = Нет , Skip\_existing: bool = True , тайм-аут: Необязательный[int] = Нет , max\_retries: Необязательный[int] = 0 )**

|  |  |
| --- | --- |
| **Параметры:** | * **выходной\_путь** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Выходной путь для записи медиафайла. Если он не указан, по умолчанию используется текущий рабочий каталог. * **имя файла** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Имя выходного файла (только основа) для записи медиафайла. Если оно не указано, используется имя файла по умолчанию. * **filename\_prefix** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Строка, которая будет добавлена ​​к имени файла. Например номер в плейлисте или название сериала. Если он не указан, ничего не будет добавлено. Это отдельно от имени файла, поэтому вы можете использовать имя файла по умолчанию, но при этом добавлять префикс. * **Skip\_existing** ( *[bool](https://docs.python.org/3/library/functions.html" \l "bool" \o "(в Python v3.11))* ) – (необязательно) Пропускать существующие файлы, по умолчанию — True. * **timeout** ( *[int](https://docs.python.org/3/library/functions.html" \l "int" \o "(в Python v3.11))* ) – (необязательно) Длина таймаута запроса в секундах. Использует системные настройки по умолчанию. * **max\_retries** ( *[int](https://docs.python.org/3/library/functions.html" \l "int" \o "(в Python v3.11))* ) – (необязательно) Количество повторных попыток после таймаута сокета. По умолчанию 0. |

**.filesize** - Размер файла медиапотока в байтах.

**.filesize\_gb** - Размер файла медиапотока в гигабайтах.

**.filesize\_mb** - Размер файла медиапотока в мегабайтах.

**.includes\_audio\_track** - Содержит ли поток только звук.

**.includes\_video\_track** - Содержит ли поток только видео.

**.is\_adaptive** - Является ли поток DASH.

**.is\_progressive** - Является ли поток прогрессивным.

**.title** - Получить название видео

## Объект StreamQuery

**pytube.query.StreamQuery( fmt\_streams ) -** Интерфейс для запроса доступных медиапотоков.

**.all( )  → Список[pytube.streams.Stream]** - Получите все результаты, представленные этим запросом, в виде списка.

**.asc( ) → pytube.query.StreamQuery** - Отсортируйте потоки в порядке возрастания.

**.count( значение: Необязательный[str] = Нет ) → int - Получите количество элементов в списке.**

**.desc( ) → pytube.query.StreamQuery - Отсортируйте потоки в порядке убывания.**

**filter( fps=Нет , разрешение=Нет , разрешение=Нет , mime\_type=Нет , тип=Нет , подтип=Нет , file\_extension=Нет , abr=Нет , битрейт=Нет , video\_codec=Нет , audio\_codec=Нет , only\_audio=Нет , only\_video=Нет , прогрессивный=Нет , адаптивный=Нет , is\_dash=Нет , custom\_filter\_functions=Нет ) -** Примените заданный критерий фильтрации.

|  |  |
| --- | --- |
| **Параметры:** | * **fps** ( *[int](https://docs.python.org/3/library/functions.html" \l "int" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Количество кадров в секунду. * **разрешение** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) псевдоним **res**. * **res** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Разрешение видео. * **mime\_type** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Двухчастный идентификатор форматов файлов и содержимого формата, состоящий из «типа» и «подтипа». * **тип** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Введите часть **mime\_type**(например, аудио, видео). * **подтип** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) часть подтипа **mime\_type**(например: mp4, mov). * **file\_extension** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Псевдоним **sub\_type**. * **abr** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Средняя скорость передачи данных (ABR) относится к среднему объему данных, передаваемых за единицу времени (например: 64 кбит/с, 192 кбит/с). * **битрейт** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) псевдоним **abr**. * **video\_codec** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Формат сжатия видео. * **audio\_codec** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Формат сжатия звука. * **Progressive** ( *[bool](https://docs.python.org/3/library/functions.html" \l "bool" \o "(в Python v3.11))* ) — исключает адаптивные потоки (один файл содержит как аудио, так и видеодорожки). * **Adaptive** ( *[bool](https://docs.python.org/3/library/functions.html" \l "bool" \o "(в Python v3.11))* ) — исключает прогрессивные потоки (аудио и видео находятся на отдельных дорожках). * **is\_dash** ( *[bool](https://docs.python.org/3/library/functions.html" \l "bool" \o "(в Python v3.11))* ) – включить/исключить потоки тире. * **only\_audio** ( *[bool](https://docs.python.org/3/library/functions.html" \l "bool" \o "(в Python v3.11))* ) – исключает потоки с видеодорожками. * **only\_video** ( *[bool](https://docs.python.org/3/library/functions.html" \l "bool" \o "(в Python v3.11))* ) – исключает потоки со звуковыми дорожками. * **custom\_filter\_functions** ( *[list](https://docs.python.org/3/library/stdtypes.html" \l "list" \o "(в Python v3.11)) или [None](https://docs.python.org/3/library/constants.html" \l "None" \o "(в Python v3.11))* ) – (необязательно) Интерфейс для определения сложных фильтров без создания подклассов. |

**first( ) → Необязательно[pytube.streams.Stream] -** Получите первые **Stream**результаты.

(первый результат этого запроса или None, если результат не содержит потоков.)

**get\_audio\_only**( подтип: str = 'mp4' ) → Необязательный[pytube.streams.Stream] - Получить аудиопоток с наивысшим битрейтом для данного кодека (по умолчанию mp4)

**get\_by\_itag**( itag: int ) → Необязательно[pytube.streams.Stream] - Получите соответствующее значение Streamдля данного itag.

**get\_by\_resolution**( разрешение: str ) → Необязательно[pytube.streams.Stream] - Получите соответствующее Streamдля данного разрешения.

**Поток должен быть в формате mp4 с прогрессивной разверткой**.

**Параметры:** разрешение( str ) — разрешение видео, например «720p», «480p», «360p», «240p», «144p».

**get\_highest\_resolution( ) → Необязательно[pytube.streams.Stream] - Получите поток самого высокого разрешения, представляющий собой прогрессивное видео.**

**get\_lowest\_resolution( )** → Необязательно[pytube.streams.Stream] - Получите поток с самым низким разрешением в формате mp4 с прогрессивной разверткой.

**last( )** - Получите последнее Streamв результатах.

**order\_by(** имя\_атрибута: str ) → pytube.query.StreamQuery - Примените порядок сортировки. Отфильтровывает поток, не имеющий атрибута.

## Объект подписи

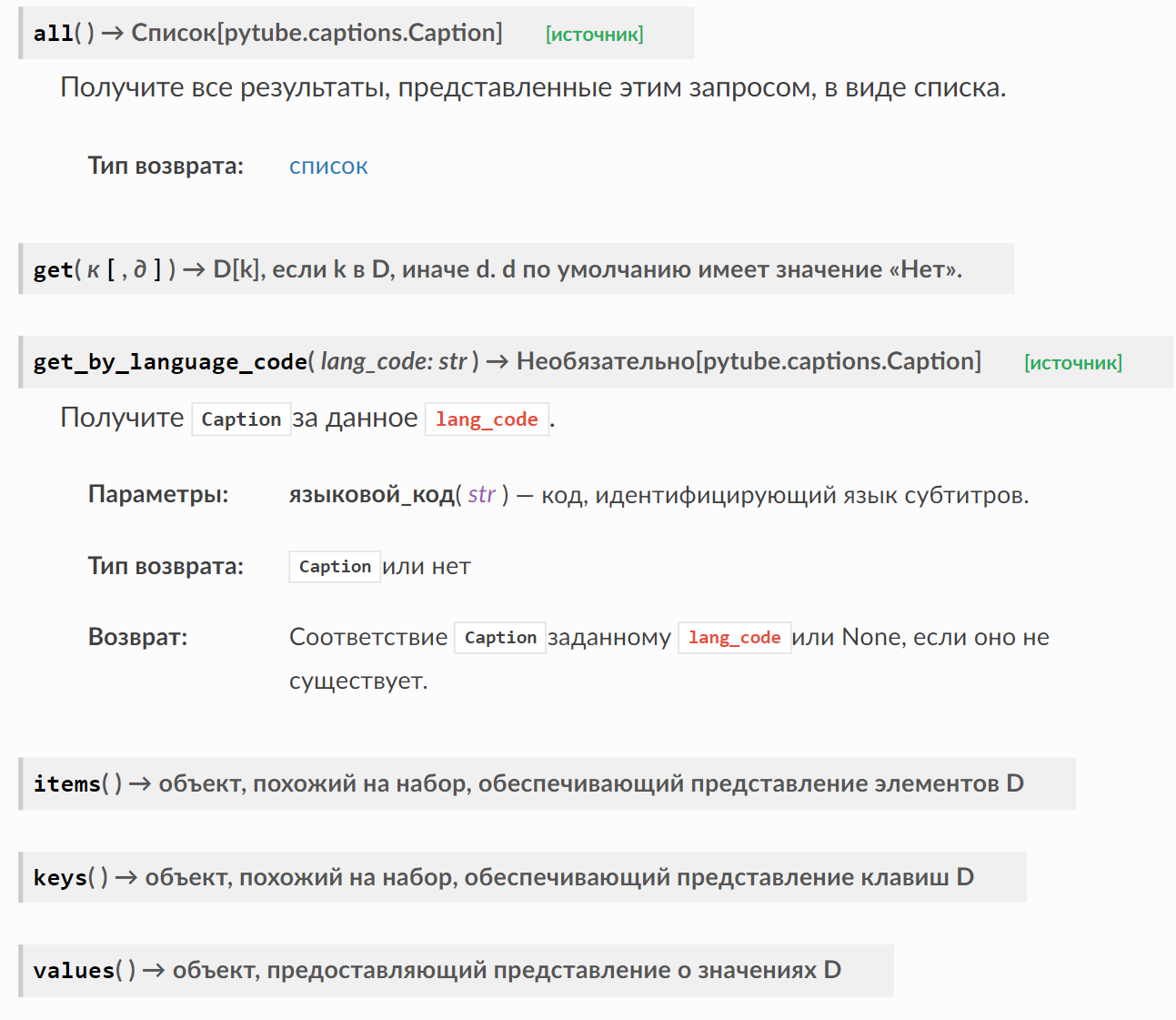
**pytube.Caption( *caption\_track: Dict[KT, VT]*) -** Контейнер для дорожек субтитров.

**download( *title: str* , *srt: bool = True* , *выходной\_путь: необязательный[str] = нет* , *префикс имени\_файла: необязательный[str] = нет*) → str -** Запишите медиапоток на диск.

|  |  |
| --- | --- |
| **Параметры:** | * **title** ( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11))* ) – имя выходного файла (только основа) для записи медиафайла. Если оно не указано, используется имя файла по умолчанию. * **srt** — установите значение True для загрузки srt и значение false для загрузки XML. По умолчанию установлено значение «Истина». |

## Объект CaptionQuery

**pytube.query.CaptionQuery( *подписи: List[pytube.captions.Caption]* -**  Интерфейс для запроса доступных подписей.



## Исключения

Определения исключений, специфичные для библиотеки.

**исключениеpytube.exceptions.AgeRestrictedError( video\_id: ул )**[**[источник]**](https://pytube.io/en/latest/_modules/pytube/exceptions.html#AgeRestrictedError)

Видео имеет возрастные ограничения, и доступ к нему невозможен без OAuth.

**исключениеpytube.exceptions.ExtractError[[источник]](https://pytube.io/en/latest/_modules/pytube/exceptions.html" \l "ExtractError)**

Исключение, основанное на извлечении данных.

**исключениеpytube.exceptions.HTMLParseError[[источник]](https://pytube.io/en/latest/_modules/pytube/exceptions.html" \l "HTMLParseError)**

HTML не удалось разобрать

**исключениеpytube.exceptions.LiveStreamError( video\_id: ул )**[**[источник]**](https://pytube.io/en/latest/_modules/pytube/exceptions.html#LiveStreamError)

Видео — это прямой эфир.

**исключениеpytube.exceptions.MaxRetriesExceeded[[источник]](https://pytube.io/en/latest/_modules/pytube/exceptions.html" \l "MaxRetriesExceeded)**

Превышено максимальное количество повторов.

**исключениеpytube.exceptions.MembersOnly( video\_id: ул )**[**[источник]**](https://pytube.io/en/latest/_modules/pytube/exceptions.html#MembersOnly)

Видео доступно только для членов.

На YouTube есть специальные видеоролики, которые могут просматривать только пользователи, подписавшиеся на создателя контента. ссылка: [https://support.google.com/youtube/answer/7544492?hl=en .](https://support.google.com/youtube/answer/7544492?hl=en)

**исключениеpytube.exceptions.PytubeError[[источник]](https://pytube.io/en/latest/_modules/pytube/exceptions.html" \l "PytubeError)**

Базовое исключение pytube, которое наследуют все остальные.

Это сделано для того, чтобы не загрязнять встроенные исключения, что может привести к неожиданной и неправильной обработке непреднамеренных ошибок в коде разработчиков.

**исключениеpytube.exceptions.RecordingUnavailable( video\_id: ул )**[**[источник]**](https://pytube.io/en/latest/_modules/pytube/exceptions.html#RecordingUnavailable)

**исключениеpytube.exceptions.RegexMatchError( вызывающий: str, шаблон: Union[str, Pattern[AnyStr]] )**[**[источник]**](https://pytube.io/en/latest/_modules/pytube/exceptions.html#RegexMatchError)

Шаблон регулярного выражения не вернул ни одного совпадения.

**исключениеpytube.exceptions.VideoPrivate( video\_id: ул )**[**[источник]**](https://pytube.io/en/latest/_modules/pytube/exceptions.html#VideoPrivate)

**исключениеpytube.exceptions.VideoRegionBlocked( video\_id: ул )**[**[источник]**](https://pytube.io/en/latest/_modules/pytube/exceptions.html#VideoRegionBlocked)

**исключениеpytube.exceptions.VideoUnavailable( video\_id: ул )**[**[источник]**](https://pytube.io/en/latest/_modules/pytube/exceptions.html#VideoUnavailable)

Ошибка «Базовое видео недоступно».

## Запрос

**pytube.request.filesize[[источник]](https://pytube.io/en/latest/_modules/pytube/request.html" \l "filesize)**

Получить размер файла в байтах по заданному URL-адресу

|  |  |
| --- | --- |
| **Параметры:** | **URL**( *[str](https://docs.python.org/3/library/stdtypes.html" \l "str" \o "(в Python v3.11))* ) – URL-адрес для получения размера |
| **Возврат:** | int: размер удаленного файла в байтах |