ytdl-org / **youtube-dl** 民众

Command-line program to download videos from YouTube.com and other video sites

ytdl-org.github.io/youtube-dl/

Unlicense License

108k stars **7.5k** forks

Star Notifications

代码 问题 4k 拉取请求 909 行动 项目 2 安全 见解

掌握 转到文件

lihan7 和 **dirkf** ··· 3天前

View code

CI passing

youtube-dl - 从 youtube.com 或其他视频平台下载视频

安装

要立即为所有 UNIX 用户（Linux、macOS 等）安装它，请键入：

```
sudo curl -L https://yt-dl.org/downloads/latest/youtube-dl -o
/usr/local/bin/youtube-dl
sudo chmod a+rx /usr/local/bin/youtube-dl
```

如果你没有 curl，你也可以使用最近的 wget：

```
sudo wget https://yt-dl.org/downloads/latest/youtube-dl -O
/usr/local/bin/youtube-dl
sudo chmod a+rx /usr/local/bin/youtube-dl
```

Windows 用户可以下载 .exe 文件并将其放置在PATH上的任何位置，除了 `%SYSTEMROOT%\System32` （例如，**不要**放入 `C:\Windows\System32` ）。

您还可以使用 pip：

```
sudo -H pip install --upgrade youtube-dl
```

如果您已安装此命令，它将更新 youtube-dl。有关更多信息，请参阅pypi 页面。

macOS 用户可以使用Homebrew安装 youtube-dl：

```
brew install youtube-dl
```

或使用MacPorts：

```
sudo port install youtube-dl
```

或者，请参阅开发人员说明，了解如何签出和使用 git 存储库。有关更多选项，包括 PGP 签名，请参阅youtube-dl 下载页面。

描述

youtube-dl是一个命令行程序，用于从 YouTube.com 和其他一些网站下载视频。它需要 Python 解释器，版本 2.6、2.7 或 3.2+，并且它不是特定于平台的。它应该可以在您的 Unix 机器、Windows 或 macOS 上运行。它已发布到公共领域，这意味着您可以修改、重新分发或随意使用它。

```
youtube-dl [OPTIONS] URL [URL...]
```

选项

```
-h, --help                       Print this help text and exit
--version                        Print program version and exit
-U, --update                     Update this program to latest version.
                                 Make sure that you have sufficient
                                 permissions (run with sudo if needed)
-i, --ignore-errors              Continue on download errors, for
                                 example to skip unavailable videos in
a
                                 playlist
--abort-on-error                 Abort downloading of further videos
(in
                                 the playlist or the command line) if
an
                                 error occurs
--dump-user-agent                Display the current browser
                                 identification
--list-extractors                List all supported extractors
--extractor-descriptions         Output descriptions of all supported
                                 extractors
--force-generic-extractor        Force extraction to use the generic
                                 extractor
--default-search PREFIX          Use this prefix for unqualified URLs.
                                 For example "gvsearch2:" downloads two
                                 videos from google videos for youtube-
                                 dl "large apple". Use the value "auto"
                                 to let youtube-dl guess
("auto_warning"
                                 to emit a warning when guessing).
                                 "error" just throws an error. The
                                 default value "fixup_error" repairs
                                 broken URLs, but emits an error if
this
                                 is not possible instead of searching.
--ignore-config                  Do not read configuration files. When
                                 given in the global configuration file
                                 /etc/youtube-dl.conf: Do not read the
                                 user configuration in
                                 ~/.config/youtube-dl/config
                                 (%APPDATA%/youtube-dl/config.txt on
                                 Windows)
--config-location PATH           Location of the configuration file;
                                 either the path to the config or its
                                 containing directory.
--flat-playlist                  Do not extract the videos of a
                                 playlist, only list them.
--mark-watched                   Mark videos watched (YouTube only)
--no-mark-watched                Do not mark videos watched (YouTube
```

	only)
--no-color	Do not emit color codes in output

网络选项：

--proxy URL	Use the specified HTTP/HTTPS/SOCKS proxy. To enable SOCKS proxy, specify a proper scheme. For example socks5://127.0.0.1:1080/. Pass in an empty string (--proxy "") for direct connection
--socket-timeout SECONDS	Time to wait before giving up, in seconds
--source-address IP	Client-side IP address to bind to
-4, --force-ipv4	Make all connections via IPv4
-6, --force-ipv6	Make all connections via IPv6

地理限制：

--geo-verification-proxy URL	Use this proxy to verify the IP address for some geo-restricted sites. The default proxy specified by --proxy (or none, if the option is not present) is used for the actual downloading.
--geo-bypass	Bypass geographic restriction via faking X-Forwarded-For HTTP header
--no-geo-bypass	Do not bypass geographic restriction via faking X-Forwarded-For HTTP header
--geo-bypass-country CODE	Force bypass geographic restriction with explicitly provided two-letter ISO 3166-2 country code
--geo-bypass-ip-block IP_BLOCK	Force bypass geographic restriction with explicitly provided IP block in CIDR notation

视频选择：

--playlist-start NUMBER	Playlist video to start at (default is 1)

```
    --playlist-end NUMBER            Playlist video to end at (default is
                                     last)
    --playlist-items ITEM_SPEC       Playlist video items to download.
                                     Specify indices of the videos in the
                                     playlist separated by commas like: "--
                                     playlist-items 1,2,5,8" if you want to
                                     download videos indexed 1, 2, 5, 8 in
                                     the playlist. You can specify range: "
                                     --playlist-items 1-3,7,10-13", it will
                                     download the videos at index 1, 2, 3,
                                     7, 10, 11, 12 and 13.
    --match-title REGEX              Download only matching titles (regex
or
                                     caseless sub-string)
    --reject-title REGEX             Skip download for matching titles
                                     (regex or caseless sub-string)
    --max-downloads NUMBER           Abort after downloading NUMBER files
    --min-filesize SIZE              Do not download any videos smaller
than
                                     SIZE (e.g. 50k or 44.6m)
    --max-filesize SIZE              Do not download any videos larger than
                                     SIZE (e.g. 50k or 44.6m)
    --date DATE                      Download only videos uploaded in this
                                     date
    --datebefore DATE                Download only videos uploaded on or
                                     before this date (i.e. inclusive)
    --dateafter DATE                 Download only videos uploaded on or
                                     after this date (i.e. inclusive)
    --min-views COUNT                Do not download any videos with less
                                     than COUNT views
    --max-views COUNT                Do not download any videos with more
                                     than COUNT views
    --match-filter FILTER            Generic video filter. Specify any key
                                     (see the "OUTPUT TEMPLATE" for a list
                                     of available keys) to match if the key
                                     is present, !key to check if the key
is
                                     not present, key > NUMBER (like
                                     "comment_count > 12", also works with
                                     >=, <, <=, !=, =) to compare against a
                                     number, key = 'LITERAL' (like
"uploader
                                     = 'Mike Smith'", also works with !=)
to
                                     match against a string literal and &
to
                                     require multiple matches. Values which
                                     are not known are excluded unless you
                                     put a question mark (?) after the
                                     operator. For example, to only match
                                     videos that have been liked more than
                                     100 times and disliked less than 50
                                     times (or the dislike functionality is
```

	not available at the given service), but who also have a description, use --match-filter "like_count > 100 & dislike_count <? 50 & description" .
--no-playlist	Download only the video, if the URL refers to a video and a playlist.
--yes-playlist	Download the playlist, if the URL refers to a video and a playlist.
--age-limit YEARS	Download only videos suitable for the given age
--download-archive FILE	Download only videos not listed in the archive file. Record the IDs of all downloaded videos in it.
--include-ads	Download advertisements as well (experimental)

下载选项：

-r, --limit-rate RATE	Maximum download rate in bytes per second (e.g. 50K or 4.2M)
-R, --retries RETRIES	Number of retries (default is 10), or "infinite".
--fragment-retries RETRIES	Number of retries for a fragment (default is 10), or "infinite" (DASH, hlsnative and ISM)
--skip-unavailable-fragments	Skip unavailable fragments (DASH, hlsnative and ISM)
--abort-on-unavailable-fragment is	Abort downloading when some fragment not available
--keep-fragments after	Keep downloaded fragments on disk downloading is finished; fragments are erased by default
--buffer-size SIZE	Size of download buffer (e.g. 1024 or 16K) (default is 1024)
--no-resize-buffer	Do not automatically adjust the buffer size. By default, the buffer size is automatically resized from an initial value of SIZE.
--http-chunk-size SIZE	Size of a chunk for chunk-based HTTP downloading (e.g. 10485760 or 10M) (default is disabled). May be useful for bypassing bandwidth throttling imposed by a webserver (experimental)
--playlist-reverse	Download playlist videos in reverse order
--playlist-random	Download playlist videos in random order

```
--xattr-set-filesize                Set file xattribute ytdl.filesize with
                                    expected file size
--hls-prefer-native                 Use the native HLS downloader instead
                                    of ffmpeg
--hls-prefer-ffmpeg                 Use ffmpeg instead of the native HLS
                                    downloader
--hls-use-mpegts                    Use the mpegts container for HLS
                                    videos, allowing to play the video
                                    while downloading (some players may
not
                                    be able to play it)
--external-downloader COMMAND       Use the specified external downloader.
                                    Currently supports
aria2c,avconv,axel,c
                                    url,ffmpeg,httpie,wget
--external-downloader-args ARGS     Give these arguments to the external
                                    downloader
```

文件系统选项：

```
-a, --batch-file FILE               File containing URLs to download ('-'
                                    for stdin), one URL per line. Lines
                                    starting with '#', ';' or ']' are
                                    considered as comments and ignored.
--id                                Use only video ID in file name
-o, --output TEMPLATE               Output filename template, see the
                                    "OUTPUT TEMPLATE" for all the info
--output-na-placeholder PLACEHOLDER Placeholder value for unavailable meta
                                    fields in output filename template
                                    (default is "NA")
--autonumber-start NUMBER           Specify the start value for
                                    %(autonumber)s (default is 1)
--restrict-filenames                Restrict filenames to only ASCII
                                    characters, and avoid "&" and spaces
in
                                    filenames
-w, --no-overwrites                 Do not overwrite files
-c, --continue                      Force resume of partially downloaded
                                    files. By default, youtube-dl will
                                    resume downloads if possible.
--no-continue                       Do not resume partially downloaded
                                    files (restart from beginning)
--no-part                           Do not use .part files - write
directly
                                    into output file
--no-mtime                          Do not use the Last-modified header to
                                    set the file modification time
--write-description                 Write video description to a
                                    .description file
```

```
--write-info-json                Write video metadata to a .info.json
                                 file
--write-annotations              Write video annotations to a
                                 .annotations.xml file
--load-info-json FILE            JSON file containing the video
                                 information (created with the "--
write-
                                 info-json" option)
--cookies FILE                   File to read cookies from and dump
                                 cookie jar in
--cache-dir DIR                  Location in the filesystem where
                                 youtube-dl can store some downloaded
                                 information permanently. By default
                                 $XDG_CACHE_HOME/youtube-dl or
                                 ~/.cache/youtube-dl . At the moment,
                                 only YouTube player files (for videos
                                 with obfuscated signatures) are
cached,
                                 but that may change.
--no-cache-dir                   Disable filesystem caching
--rm-cache-dir                   Delete all filesystem cache files
```

缩略图选项：

```
--write-thumbnail                Write thumbnail image to disk
--write-all-thumbnails           Write all thumbnail image formats to
                                 disk
--list-thumbnails                Simulate and list all available
                                 thumbnail formats
```

详细程度/模拟选项：

```
-q, --quiet                      Activate quiet mode
--no-warnings                    Ignore warnings
-s, --simulate                   Do not download the video and do not
                                 write anything to disk
--skip-download                  Do not download the video
-g, --get-url                    Simulate, quiet but print URL
-e, --get-title                  Simulate, quiet but print title
--get-id                         Simulate, quiet but print id
--get-thumbnail                  Simulate, quiet but print thumbnail
URL
--get-description                Simulate, quiet but print video
                                 description
--get-duration                   Simulate, quiet but print video length
--get-filename                   Simulate, quiet but print output
```

```
                                  filename
--get-format                      Simulate, quiet but print output
format
-j, --dump-json                   Simulate, quiet but print JSON
                                  information. See the "OUTPUT TEMPLATE"
                                  for a description of available keys.
-J, --dump-single-json            Simulate, quiet but print JSON
                                  information for each command-line
                                  argument. If the URL refers to a
                                  playlist, dump the whole playlist
                                  information in a single line.
--print-json                      Be quiet and print the video
                                  information as JSON (video is still
                                  being downloaded).
--newline                         Output progress bar as new lines
--no-progress                     Do not print progress bar
--console-title                   Display progress in console titlebar
-v, --verbose                     Print various debugging information
--dump-pages                      Print downloaded pages encoded using
                                  base64 to debug problems (very
verbose)
--write-pages                     Write downloaded intermediary pages to
                                  files in the current directory to
debug
                                  problems
--print-traffic                   Display sent and read HTTP traffic
-C, --call-home                   Contact the youtube-dl server for
                                  debugging
--no-call-home                    Do NOT contact the youtube-dl server
                                  for debugging
```

解决方法：

```
--encoding ENCODING               Force the specified encoding
                                  (experimental)
--no-check-certificate            Suppress HTTPS certificate validation
--prefer-insecure                 Use an unencrypted connection to
                                  retrieve information about the video.
                                  (Currently supported only for YouTube)
--user-agent UA                   Specify a custom user agent
--referer URL                     Specify a custom referer, use if the
                                  video access is restricted to one
                                  domain
--add-header FIELD:VALUE          Specify a custom HTTP header and its
                                  value, separated by a colon ':'. You
                                  can use this option multiple times
--bidi-workaround                 Work around terminals that lack
                                  bidirectional text support. Requires
                                  bidiv or fribidi executable in PATH
```

```
--sleep-interval SECONDS          Number of seconds to sleep before each
                                  download when used alone or a lower
                                  bound of a range for randomized sleep
                                  before each download (minimum possible
                                  number of seconds to sleep) when used
                                  along with --max-sleep-interval.
--max-sleep-interval SECONDS      Upper bound of a range for randomized
                                  sleep before each download (maximum
                                  possible number of seconds to sleep).
                                  Must only be used along with --min-
                                  sleep-interval.
```

视频格式选项：

```
-f, --format FORMAT               Video format code, see the "FORMAT
                                  SELECTION" for all the info
--all-formats                     Download all available video formats
--prefer-free-formats             Prefer free video formats unless a
                                  specific one is requested
-F, --list-formats                List all available formats of
requested
                                  videos
--youtube-skip-dash-manifest      Do not download the DASH manifests and
                                  related data on YouTube videos
--merge-output-format FORMAT      If a merge is required (e.g.
                                  bestvideo+bestaudio), output to given
                                  container format. One of mkv, mp4,
ogg,
                                  webm, flv. Ignored if no merge is
                                  required
```

字幕选项：

```
--write-sub                       Write subtitle file
--write-auto-sub                  Write automatically generated subtitle
                                  file (YouTube only)
--all-subs                        Download all the available subtitles
of
                                  the video
--list-subs                       List all available subtitles for the
                                  video
--sub-format FORMAT               Subtitle format, accepts formats
                                  preference, for example: "srt" or
                                  "ass/srt/best"
--sub-lang LANGS                  Languages of the subtitles to download
                                  (optional) separated by commas, use
```

```
                                  --list-subs for available language
tags
```

身份验证选项：

```
-u, --username USERNAME           Login with this account ID
-p, --password PASSWORD           Account password. If this option is
                                  left out, youtube-dl will ask
                                  interactively.
-2, --twofactor TWOFACTOR         Two-factor authentication code
-n, --netrc                       Use .netrc authentication data
--video-password PASSWORD         Video password (vimeo, youku)
```

Adobe 通行证选项：

```
--ap-mso MSO                      Adobe Pass multiple-system operator
(TV
                                  provider) identifier, use --ap-list-
mso
                                  for a list of available MSOs
--ap-username USERNAME            Multiple-system operator account login
--ap-password PASSWORD            Multiple-system operator account
                                  password. If this option is left out,
                                  youtube-dl will ask interactively.
--ap-list-mso                     List all supported multiple-system
                                  operators
```

后处理选项：

```
-x, --extract-audio               Convert video files to audio-only
files
                                  (requires ffmpeg/avconv and
                                  ffprobe/avprobe)
--audio-format FORMAT             Specify audio format: "best", "aac",
                                  "flac", "mp3", "m4a", "opus",
"vorbis",
                                  or "wav"; "best" by default; No effect
                                  without -x
--audio-quality QUALITY           Specify ffmpeg/avconv audio quality,
                                  insert a value between 0 (better) and
9
                                  (worse) for VBR or a specific bitrate
```

```
                                  like 128K (default 5)
    --recode-video FORMAT         Encode the video to another format if
                                  necessary (currently supported:
                                  mp4|flv|ogg|webm|mkv|avi)
    --postprocessor-args ARGS     Give these arguments to the
                                  postprocessor
    -k, --keep-video              Keep the video file on disk after the
                                  post-processing; the video is erased
    by
                                  default
    --no-post-overwrites          Do not overwrite post-processed files;
                                  the post-processed files are
                                  overwritten by default
    --embed-subs                  Embed subtitles in the video (only for
                                  mp4, webm and mkv videos)
    --embed-thumbnail             Embed thumbnail in the audio as cover
                                  art
    --add-metadata                Write metadata to the video file
    --metadata-from-title FORMAT  Parse additional metadata like song
                                  title / artist from the video title.
                                  The format syntax is the same as
                                  --output. Regular expression with
    named
                                  capture groups may also be used. The
                                  parsed parameters replace existing
                                  values. Example: --metadata-from-title
                                  "%(artist)s - %(title)s" matches a
                                  title like "Coldplay - Paradise".
                                  Example (regex): --metadata-from-title
                                  "(?P<artist>.+?) - (?P<title>.+)"
    --xattrs                      Write metadata to the video file's
                                  xattrs (using dublin core and xdg
                                  standards)
    --fixup POLICY                Automatically correct known faults of
                                  the file. One of never (do nothing),
                                  warn (only emit a warning),
                                  detect_or_warn (the default; fix file
                                  if we can, warn otherwise)
    --prefer-avconv               Prefer avconv over ffmpeg for running
                                  the postprocessors
    --prefer-ffmpeg               Prefer ffmpeg over avconv for running
                                  the postprocessors (default)
    --ffmpeg-location PATH        Location of the ffmpeg/avconv binary;
                                  either the path to the binary or its
                                  containing directory.
    --exec CMD                    Execute a command on the file after
                                  downloading and post-processing,
                                  similar to find's -exec syntax.
                                  Example: --exec 'adb push {}
                                  /sdcard/Music/ && rm {}'
    --convert-subs FORMAT         Convert the subtitles to other format
                                  (currently supported: srt|ass|vtt|lrc)
```

配置

您可以通过将任何受支持的命令行选项放入配置文件来配置 youtube-dl。在 Linux 和 macOS 上，系统范围的配置文件位于， `/etc/youtube-dl.conf` 用户范围的配置文件位于 `~/.config/youtube-dl/config` 。在 Windows 上，用户范围的配置文件位置是 `%APPDATA%\youtube-dl\config.txt` 或 `C:\Users\<user name>\youtube-dl.conf` 。请注意，默认情况下配置文件可能不存在，因此您可能需要自己创建它。

例如，使用以下配置文件 youtube-dl 将始终提取音频，而不是复制 mtime，使用代理并将所有视频保存 `Movies` 在您的主目录中的目录下：

```
# Lines starting with # are comments

# Always extract audio
-x

# Do not copy the mtime
--no-mtime

# Use this proxy
--proxy 127.0.0.1:3128

# Save all videos under Movies directory in your home directory
-o ~/Movies/%(title)s.%(ext)s
```

Note that options in configuration file are just the same options aka switches used in regular command line calls thus there **must be no whitespace** after `-` or `--` , e.g. `-o` or `--proxy` but not `- o` or `-- proxy` .

You can use `--ignore-config` if you want to disable the configuration file for a particular youtube-dl run.

You can also use `--config-location` if you want to use custom configuration file for a particular youtube-dl run.

Authentication with `.netrc` file

You may also want to configure automatic credentials storage for extractors that support authentication (by providing login and password with `--username` and `--password`) in order not to pass credentials as command line arguments on every youtube-dl execution and prevent tracking plain text passwords in the shell command history. You can achieve this using a `.netrc` file on a per extractor basis. For that you will need to create a `.netrc` file in your `$HOME` and restrict permissions to read/write by only you:

```
touch $HOME/.netrc
chmod a-rwx,u+rw $HOME/.netrc
```

After that you can add credentials for an extractor in the following format, where *extractor* is the name of the extractor in lowercase:

```
machine <extractor> login <login> password <password>
```

For example:

```
machine youtube login myaccount@gmail.com password my_youtube_password
machine twitch login my_twitch_account_name password my_twitch_password
```

To activate authentication with the `.netrc` file you should pass `--netrc` to youtube-dl or place it in the configuration file.

On Windows you may also need to setup the `%HOME%` environment variable manually. For example:

```
set HOME=%USERPROFILE%
```

OUTPUT TEMPLATE

The `-o` option allows users to indicate a template for the output file names.

tl;dr: navigate me to examples.

The basic usage is not to set any template arguments when downloading a single file, like in `youtube-dl -o funny_video.flv "https://some/video"` . However, it may contain special sequences that will be replaced when downloading each video. The special sequences may be formatted according to python string formatting operations. For example, `%(NAME)s` or `%(NAME)05d` . To clarify, that is a percent symbol followed by a name in parentheses, followed by formatting operations. Allowed names along with sequence type are:

- `id` (string): Video identifier
- `title` (string): Video title
- `url` (string): Video URL
- `ext` (string): Video filename extension
- `alt_title` (string): A secondary title of the video

- `display_id` (string): An alternative identifier for the video
- `uploader` (string): Full name of the video uploader
- `license` (string): License name the video is licensed under
- `creator` (string): The creator of the video
- `release_date` (string): The date (YYYYMMDD) when the video was released
- `timestamp` (numeric): UNIX timestamp of the moment the video became available
- `upload_date` (string): Video upload date (YYYYMMDD)
- `uploader_id` (string): Nickname or id of the video uploader
- `channel` (string): Full name of the channel the video is uploaded on
- `channel_id` (string): Id of the channel
- `location` (string): Physical location where the video was filmed
- `duration` (numeric): Length of the video in seconds
- `view_count` (numeric): How many users have watched the video on the platform
- `like_count` (numeric): Number of positive ratings of the video
- `dislike_count` (numeric): Number of negative ratings of the video
- `repost_count` (numeric): Number of reposts of the video
- `average_rating` (numeric): Average rating give by users, the scale used depends on the webpage
- `comment_count` (numeric): Number of comments on the video
- `age_limit` (numeric): Age restriction for the video (years)
- `is_live` (boolean): Whether this video is a live stream or a fixed-length video
- `start_time` (numeric): Time in seconds where the reproduction should start, as specified in the URL
- `end_time` (numeric): Time in seconds where the reproduction should end, as specified in the URL
- `format` (string): A human-readable description of the format
- `format_id` (string): Format code specified by `--format`
- `format_note` (string): Additional info about the format
- `width` (numeric): Width of the video
- `height` (numeric): Height of the video
- `resolution` (string): Textual description of width and height
- `tbr` (numeric): Average bitrate of audio and video in KBit/s
- `abr` (numeric): Average audio bitrate in KBit/s
- `acodec` (string): Name of the audio codec in use
- `asr` (numeric): Audio sampling rate in Hertz
- `vbr` (numeric): Average video bitrate in KBit/s
- `fps` (numeric): Frame rate

- `vcodec` (string): Name of the video codec in use
- `container` (string): Name of the container format
- `filesize` (numeric): The number of bytes, if known in advance
- `filesize_approx` (numeric): An estimate for the number of bytes
- `protocol` (string): The protocol that will be used for the actual download
- `extractor` (string): Name of the extractor
- `extractor_key` (string): Key name of the extractor
- `epoch` (numeric): Unix epoch when creating the file
- `autonumber` (numeric): Number that will be increased with each download, starting at `--autonumber-start`
- `playlist` (string): Name or id of the playlist that contains the video
- `playlist_index` (numeric): Index of the video in the playlist padded with leading zeros according to the total length of the playlist
- `playlist_id` (string): Playlist identifier
- `playlist_title` (string): Playlist title
- `playlist_uploader` (string): Full name of the playlist uploader
- `playlist_uploader_id` (string): Nickname or id of the playlist uploader

Available for the video that belongs to some logical chapter or section:

- `chapter` (string): Name or title of the chapter the video belongs to
- `chapter_number` (numeric): Number of the chapter the video belongs to
- `chapter_id` (string): Id of the chapter the video belongs to

Available for the video that is an episode of some series or programme:

- `series` (string): Title of the series or programme the video episode belongs to
- `season` (string): Title of the season the video episode belongs to
- `season_number` (numeric): Number of the season the video episode belongs to
- `season_id` (string): Id of the season the video episode belongs to
- `episode` (string): Title of the video episode
- `episode_number` (numeric): Number of the video episode within a season
- `episode_id` (string): Id of the video episode

Available for the media that is a track or a part of a music album:

- `track` (string): Title of the track
- `track_number` (numeric): Number of the track within an album or a disc
- `track_id` (string): Id of the track
- `artist` (string): Artist(s) of the track

- `genre` (string): Genre(s) of the track
- `album` (string): Title of the album the track belongs to
- `album_type` (string): Type of the album
- `album_artist` (string): List of all artists appeared on the album
- `disc_number` (numeric): Number of the disc or other physical medium the track belongs to
- `release_year` (numeric): Year (YYYY) when the album was released

Each aforementioned sequence when referenced in an output template will be replaced by the actual value corresponding to the sequence name. Note that some of the sequences are not guaranteed to be present since they depend on the metadata obtained by a particular extractor. Such sequences will be replaced with placeholder value provided with `--output-na-placeholder` (`NA` by default).

For example for `-o %(title)s-%(id)s.%(ext)s` and an mp4 video with title `youtube-dl test video` and id `BaW_jenozKcj`, this will result in a `youtube-dl test video-BaW_jenozKcj.mp4` file created in the current directory.

For numeric sequences you can use numeric related formatting, for example, `%(view_count)05d` will result in a string with view count padded with zeros up to 5 characters, like in `00042`.

Output templates can also contain arbitrary hierarchical path, e.g. `-o '%(playlist)s/%(playlist_index)s - %(title)s.%(ext)s'` which will result in downloading each video in a directory corresponding to this path template. Any missing directory will be automatically created for you.

To use percent literals in an output template use `%%`. To output to stdout use `-o -`.

The current default template is `%(title)s-%(id)s.%(ext)s`.

In some cases, you don't want special characters such as 中, spaces, or &, such as when transferring the downloaded filename to a Windows system or the filename through an 8bit-unsafe channel. In these cases, add the `--restrict-filenames` flag to get a shorter title:

Output template and Windows batch files

If you are using an output template inside a Windows batch file then you must escape plain percent characters (`%`) by doubling, so that `-o "%(title)s-%(id)s.%(ext)s"` should become `-o "%%(title)s-%%(id)s.%%(ext)s"`. However you should not touch `%`'s that are not plain characters, e.g. environment variables for expansion should stay intact: `-o "C:\%HOMEPATH%\Desktop\%%(title)s.%%(ext)s"`.

Output template examples

Note that on Windows you may need to use double quotes instead of single.

```
$ youtube-dl --get-filename -o '%(title)s.%(ext)s' BaW_jenozKc
youtube-dl test video ''_ä↭𝕐.mp4    # All kinds of weird characters

$ youtube-dl --get-filename -o '%(title)s.%(ext)s' BaW_jenozKc --restrict-file
youtube-dl_test_video_.mp4          # A simple file name

# Download YouTube playlist videos in separate directory indexed by video orde
$ youtube-dl -o '%(playlist)s/%(playlist_index)s - %(title)s.%(ext)s' https://

# Download all playlists of YouTube channel/user keeping each playlist in sepa
$ youtube-dl -o '%(uploader)s/%(playlist)s/%(playlist_index)s - %(title)s.%(ex

# Download Udemy course keeping each chapter in separate directory under MyVid
$ youtube-dl -u user -p password -o '~/MyVideos/%(playlist)s/%(chapter_number)

# Download entire series season keeping each series and each season in separat
$ youtube-dl -o "C:/MyVideos/%(series)s/%(season_number)s - %(season)s/%(episo

# Stream the video being downloaded to stdout
$ youtube-dl -o - BaW_jenozKc
```

FORMAT SELECTION

By default youtube-dl tries to download the best available quality, i.e. if you want the best quality you **don't need** to pass any special options, youtube-dl will guess it for you by **default**.

But sometimes you may want to download in a different format, for example when you are on a slow or intermittent connection. The key mechanism for achieving this is so-called *format selection* based on which you can explicitly specify desired format, select formats based on some criterion or criteria, setup precedence and much more.

The general syntax for format selection is `--format FORMAT` or shorter `-f FORMAT` where `FORMAT` is a *selector expression*, i.e. an expression that describes format or formats you would like to download.

tl;dr: navigate me to examples.

The simplest case is requesting a specific format, for example with `-f 22` you can download the format with format code equal to 22. You can get the list of available format codes for particular video using `--list-formats` or `-F`. Note that these format codes are extractor specific.

You can also use a file extension (currently `3gp`, `aac`, `flv`, `m4a`, `mp3`, `mp4`, `ogg`, `wav`, `webm` are supported) to download the best quality format of a particular file extension served as a single file, e.g. `-f webm` will download the best quality format with the `webm` extension served as a single file.

You can also use special names to select particular edge case formats:

- `best`: Select the best quality format represented by a single file with video and audio.
- `worst`: Select the worst quality format represented by a single file with video and audio.
- `bestvideo`: Select the best quality video-only format (e.g. DASH video). May not be available.
- `worstvideo`: Select the worst quality video-only format. May not be available.
- `bestaudio`: Select the best quality audio only-format. May not be available.
- `worstaudio`: Select the worst quality audio only-format. May not be available.

For example, to download the worst quality video-only format you can use `-f worstvideo`.

If you want to download multiple videos and they don't have the same formats available, you can specify the order of preference using slashes. Note that slash is left-associative, i.e. formats on the left hand side are preferred, for example `-f 22/17/18` will download format 22 if it's available, otherwise it will download format 17 if it's available, otherwise it will download format 18 if it's available, otherwise it will complain that no suitable formats are available for download.

If you want to download several formats of the same video use a comma as a separator, e.g. `-f 22,17,18` will download all these three formats, of course if they are available. Or a more sophisticated example combined with the precedence feature: `-f 136/137/mp4/bestvideo,140/m4a/bestaudio`.

You can also filter the video formats by putting a condition in brackets, as in `-f "best[height=720]"` (or `-f "[filesize>10M]"`).

The following numeric meta fields can be used with comparisons `<`, `<=`, `>`, `>=`, `=` (equals), `!=` (not equals):

- `filesize`: The number of bytes, if known in advance
- `width`: Width of the video, if known
- `height`: Height of the video, if known
- `tbr`: Average bitrate of audio and video in KBit/s
- `abr`: Average audio bitrate in KBit/s
- `vbr`: Average video bitrate in KBit/s
- `asr`: Audio sampling rate in Hertz

- `fps` : Frame rate

Also filtering work for comparisons `=` (equals), `^=` (starts with), `$=` (ends with), `*=` (contains) and following string meta fields:

- `ext` : File extension
- `acodec` : Name of the audio codec in use
- `vcodec` : Name of the video codec in use
- `container` : Name of the container format
- `protocol` : The protocol that will be used for the actual download, lower-case (`http` , `https` , `rtsp` , `rtmp` , `rtmpe` , `mms` , `f4m` , `ism` , `http_dash_segments` , `m3u8` , or `m3u8_native`)
- `format_id` : A short description of the format
- `language` : Language code

Any string comparison may be prefixed with negation `!` in order to produce an opposite comparison, e.g. `!*=` (does not contain).

Note that none of the aforementioned meta fields are guaranteed to be present since this solely depends on the metadata obtained by particular extractor, i.e. the metadata offered by the video hoster.

Formats for which the value is not known are excluded unless you put a question mark (`?`) after the operator. You can combine format filters, so `-f "[height <=? 720][tbr>500]"` selects up to 720p videos (or videos where the height is not known) with a bitrate of at least 500 KBit/s.

You can merge the video and audio of two formats into a single file using `-f <video-format>+<audio-format>` (requires ffmpeg or avconv installed), for example `-f bestvideo+bestaudio` will download the best video-only format, the best audio-only format and mux them together with ffmpeg/avconv.

Format selectors can also be grouped using parentheses, for example if you want to download the best mp4 and webm formats with a height lower than 480 you can use `-f '(mp4,webm)[height<480]'` .

Since the end of April 2015 and version 2015.04.26, youtube-dl uses `-f bestvideo+bestaudio/best` as the default format selection (see #5447, #5456). If ffmpeg or avconv are installed this results in downloading `bestvideo` and `bestaudio` separately and muxing them together into a single file giving the best overall quality available. Otherwise it falls back to `best` and results in downloading the best available quality served as a single file. `best` is also needed for videos that don't come from YouTube because they don't provide the audio and video in two different files. If you want to only download some DASH formats (for example if you are not interested in getting videos with a resolution higher than 1080p), you can add `-f bestvideo[height<=?1080]+bestaudio/best` to your configuration file. Note that if you use youtube-dl to stream to `stdout` (and most likely to pipe it to your media player then), i.e. you explicitly specify output template as `-o -`, youtube-dl still uses `-f best` format selection in order to start content delivery immediately to your player and not to wait until `bestvideo` and `bestaudio` are downloaded and muxed.

If you want to preserve the old format selection behavior (prior to youtube-dl 2015.04.26), i.e. you want to download the best available quality media served as a single file, you should explicitly specify your choice with `-f best`. You may want to add it to the configuration file in order not to type it every time you run youtube-dl.

Format selection examples

Note that on Windows you may need to use double quotes instead of single.

```
# Download best mp4 format available or any other best if no mp4 available
$ youtube-dl -f 'bestvideo[ext=mp4]+bestaudio[ext=m4a]/best[ext=mp4]/best'

# Download best format available but no better than 480p
$ youtube-dl -f 'bestvideo[height<=480]+bestaudio/best[height<=480]'

# Download best video only format but no bigger than 50 MB
$ youtube-dl -f 'best[filesize<50M]'

# Download best format available via direct link over HTTP/HTTPS protocol
$ youtube-dl -f '(bestvideo+bestaudio/best)[protocol^=http]'

# Download the best video format and the best audio format without merging the
$ youtube-dl -f 'bestvideo,bestaudio' -o '%(title)s.f%(format_id)s.%(ext)s'
```

Note that in the last example, an output template is recommended as bestvideo and bestaudio may have the same file name.

VIDEO SELECTION

Videos can be filtered by their upload date using the options `--date`, `--datebefore` or `--dateafter`. They accept dates in two formats:

- Absolute dates: Dates in the format `YYYYMMDD`.
- Relative dates: Dates in the format `(now|today)[+-][0-9](day|week|month|year)(s)?`

Examples:

```
# Download only the videos uploaded in the last 6 months
$ youtube-dl --dateafter now-6months

# Download only the videos uploaded on January 1, 1970
$ youtube-dl --date 19700101

$ # Download only the videos uploaded in the 200x decade
$ youtube-dl --dateafter 20000101 --datebefore 20091231
```

FAQ

How do I update youtube-dl?

If you've followed our manual installation instructions, you can simply run `youtube-dl -U` (or, on Linux, `sudo youtube-dl -U`).

If you have used pip, a simple `sudo pip install -U youtube-dl` is sufficient to update.

If you have installed youtube-dl using a package manager like *apt-get* or *yum*, use the standard system update mechanism to update. Note that distribution packages are often outdated. As a rule of thumb, youtube-dl releases at least once a month, and often weekly or even daily. Simply go to https://yt-dl.org to find out the current version. Unfortunately, there is nothing we youtube-dl developers can do if your distribution serves a really outdated version. You can (and should) complain to your distribution in their bugtracker or support forum.

As a last resort, you can also uninstall the version installed by your package manager and follow our manual installation instructions. For that, remove the distribution's package, with a line like

```
sudo apt-get remove -y youtube-dl
```

Afterwards, simply follow our manual installation instructions:

```
sudo wget https://yt-dl.org/downloads/latest/youtube-dl -O
/usr/local/bin/youtube-dl
sudo chmod a+rx /usr/local/bin/youtube-dl
hash -r
```

Again, from then on you'll be able to update with `sudo youtube-dl -U`.

youtube-dl is extremely slow to start on Windows

Add a file exclusion for `youtube-dl.exe` in Windows Defender settings.

I'm getting an error `Unable to extract OpenGraph title` on YouTube playlists

YouTube changed their playlist format in March 2014 and later on, so you'll need at least youtube-dl 2014.07.25 to download all YouTube videos.

If you have installed youtube-dl with a package manager, pip, setup.py or a tarball, please use that to update. Note that Ubuntu packages do not seem to get updated anymore. Since we are not affiliated with Ubuntu, there is little we can do. Feel free to report bugs to the Ubuntu packaging people - all they have to do is update the package to a somewhat recent version. See above for a way to update.

I'm getting an error when trying to use output template: `error: using output template conflicts with using title, video ID or auto number`

Make sure you are not using `-o` with any of these options `-t`, `--title`, `--id`, `-A` or `--auto-number` set in command line or in a configuration file. Remove the latter if any.

Do I always have to pass `-citw`?

By default, youtube-dl intends to have the best options (incidentally, if you have a convincing case that these should be different, please file an issue where you explain that). Therefore, it is unnecessary and sometimes harmful to copy long option strings from webpages. In particular, the only option out of `-citw` that is regularly useful is `-i`.

Can you please put the `-b` option back?

Most people asking this question are not aware that youtube-dl now defaults to downloading the highest available quality as reported by YouTube, which will be 1080p or 720p in some cases, so you no longer need the `-b` option. For some specific videos, maybe YouTube does not report them to be available in a specific high quality format you're interested in. In that case, simply request it with the `-f` option and youtube-dl will try to download it.

I get HTTP error 402 when trying to download a video. What's this?

Apparently YouTube requires you to pass a CAPTCHA test if you download too much. We're considering to provide a way to let you solve the CAPTCHA, but at the moment, your best course of action is pointing a web browser to the youtube URL, solving the CAPTCHA, and restart youtube-dl.

Do I need any other programs?

youtube-dl works fine on its own on most sites. However, if you want to convert video/audio, you'll need avconv or ffmpeg. On some sites - most notably YouTube - videos can be retrieved in a higher quality format without sound. youtube-dl will detect whether avconv/ffmpeg is present and automatically pick the best option.

Videos or video formats streamed via RTMP protocol can only be downloaded when rtmpdump is installed. Downloading MMS and RTSP videos requires either mplayer or mpv to be installed.

I have downloaded a video but how can I play it?

Once the video is fully downloaded, use any video player, such as mpv, vlc or mplayer.

I extracted a video URL with `-g`, but it does not play on another machine / in my web browser.

It depends a lot on the service. In many cases, requests for the video (to download/play it) must come from the same IP address and with the same cookies and/or HTTP headers. Use the `--cookies` option to write the required cookies into a file, and advise your downloader to read cookies from that file. Some sites also require a common user agent to be used, use `--dump-user-agent` to see the one in use by youtube-dl. You can also get necessary cookies and HTTP headers from JSON output obtained with `--dump-json`.

It may be beneficial to use IPv6; in some cases, the restrictions are only applied to IPv4. Some services (sometimes only for a subset of videos) do not restrict the video URL by IP address, cookie, or user-agent, but these are the exception rather than the rule.

请记住，某些 URL 协议**不受**开箱即用的浏览器支持，包括 RTMP。如果您正在使用 `-g`，您自己的下载器也必须支持这些。

如果您想在没有运行 youtube-dl 的机器上播放视频，您可以从运行 youtube-dl 的机器中继视频内容。您可以使用 `-o -` 让 youtube-dl 将视频流式传输到标准输出，或者只是让播放器依次下载 youtube-dl 编写的文件。

错误：在视频信息中找不到 fmt_url_map 或 conn 信息

YouTube 已于 2011 年 7 月切换到新的视频信息格式，旧版本的 youtube-dl 不支持这种格式。有关如何更新 youtube-dl 的信息，请参见上文。

错误：无法下载视频

自 2012 年 9 月起，YouTube 需要额外的签名，旧版本的 youtube-dl 不支持该签名。有关如何更新 youtube-dl 的信息，请参见上文。

视频 URL 包含一个 & 符号，我得到一些奇怪的输出 [1] 2839 或 'v' is not recognized as an internal or external command

这实际上是你的 shell 的输出。由于 & 是特殊的 shell 字符之一，它由 shell 解释，阻止您将整个 URL 传递给 youtube-dl。要禁止您的 shell 解释与符号（或任何其他特殊字符），您必须将整个 URL 放在引号中或用反斜杠转义它们（哪种方法有效取决于您的 shell）。

例如，如果您的网址是https://www.youtube.com/watch?t=4&v=BaW_jenozKc，您应该以以下命令结束：

```
youtube-dl 'https://www.youtube.com/watch?t=4&v=BaW_jenozKc'
```

要么

```
youtube-dl https://www.youtube.com/watch?t=4\&v=BaW_jenozKc
```

对于 Windows，您必须使用双引号：

```
youtube-dl "https://www.youtube.com/watch?t=4&v=BaW_jenozKc"
```

ExtractorError: 找不到 JS 函数 u'OF'

2015 年 2 月，新的 YouTube 播放器在字符串中包含一个被旧版本 youtube-dl 误解的字符序列。有关如何更新 youtube-dl 的信息，请参见上文。

HTTP 错误 429：请求过多或 402：需要付款

这两个错误代码表明该服务由于过度使用而阻止了您的 IP 地址。通常这是一个软块，意味着您可以在解决 CAPTCHA 后再次获得访问权限。只需打开浏览器并解决服务建议的验证码，然后将 cookie 传递给 youtube-dl。请注意，如果您的机器有多个外部 IP，那么您还应该传递与解决 CAPTCHA 相同的 IP `--source-address` 。此外，您可能需要将 `User-Agent` 浏览器的 HTTP 标头与 `--user-agent` .

如果不是这种情况（服务没有建议解决验证码），那么您可以联系服务并要求他们解除对您的 IP 地址的阻止，或者 - 如果您已经获得了列入白名单的 IP 地址 - 使用 `--proxy` 或 `--source-address` 选项来选择另一个IP地址。

语法错误：非 ASCII 字符

错误

```
  File "youtube-dl", line 2
  SyntaxError: Non-ASCII character '\x93' ...
```

意味着您使用的是过时的 Python 版本。请更新到 Python 2.6 或 2.7。

这个二进制文件是什么？代码哪里去了？

自 2012 年 6 月 (#342) 起，youtube-dl 被打包为可执行的 zip 文件，只需解压缩它（在某些系统上可能需要 `youtube-dl.zip` 先重命名）或克隆 git 存储库，如上所述。如果你修改了代码，你可以通过执行 `__main__.py` 文件来运行它。要重新编译可执行文件，请运行 `make youtube-dl` .

The exe throws an error due to missing `MSVCR100.dll`

To run the exe you need to install first the Microsoft Visual C++ 2010 Service Pack 1 Redistributable Package (x86).

On Windows, how should I set up ffmpeg and youtube-dl? Where should I put the exe files?

If you put youtube-dl and ffmpeg in the same directory that you're running the command from, it will work, but that's rather cumbersome.

To make a different directory work - either for ffmpeg, or for youtube-dl, or for both - simply create the directory (say, `C:\bin` , or `C:\Users\<User name>\bin`), put all the executables directly in there, and then set your PATH environment variable to include that directory.

From then on, after restarting your shell, you will be able to access both youtube-dl and ffmpeg (and youtube-dl will be able to find ffmpeg) by simply typing `youtube-dl` or `ffmpeg` , no matter what directory you're in.

How do I put downloads into a specific folder?

Use the `-o` to specify an output template, for example `-o "/home/user/videos/%(title)s-%(id)s.%(ext)s"` . If you want this for all of your downloads, put the option into your configuration file.

How do I download a video starting with a `-` ?

Either prepend `https://www.youtube.com/watch?v=` or separate the ID from the options with `--`:

```
youtube-dl -- -wNyEUrxzFU
youtube-dl "https://www.youtube.com/watch?v=-wNyEUrxzFU"
```

How do I pass cookies to youtube-dl?

Use the `--cookies` option, for example `--cookies /path/to/cookies/file.txt`.

In order to extract cookies from browser use any conforming browser extension for exporting cookies. For example, Get cookies.txt (for Chrome) or cookies.txt (for Firefox).

Note that the cookies file must be in Mozilla/Netscape format and the first line of the cookies file must be either `# HTTP Cookie File` or `# Netscape HTTP Cookie File`. Make sure you have correct newline format in the cookies file and convert newlines if necessary to correspond with your OS, namely `CRLF` (`\r\n`) for Windows and `LF` (`\n`) for Unix and Unix-like systems (Linux, macOS, etc.). `HTTP Error 400: Bad Request` when using `--cookies` is a good sign of invalid newline format.

当特定提取器未明确实现它时，将 cookie 传递给 youtube-dl 是一种解决登录问题的好方法。另一个用例是围绕CAPTCHA工作，一些网站需要您在特定情况下解决才能获得访问权限（例如 YouTube、CloudFlare）。

如何直接流式传输到媒体播放器?

您首先需要告诉 youtube-dl 将媒体流式传输到标准输出 `-o -`，并告诉您的媒体播放器从标准输入读取（它必须能够进行流式传输），然后将前者传输到后者。例如，可以通过以下方式实现流式传输到vlc：

```
youtube-dl -o - "https://www.youtube.com/watch?v=BaW_jenozKcj" | vlc -
```

如何仅从播放列表下载新视频?

使用下载存档功能。使用此功能，您应该首先下载完整的播放列表，`--download-archive /path/to/download/archive/file.txt` 其中会将所有视频的标识符记录在一个特殊文件中。每次后续运行 `--download-archive` 都将只下载新视频并跳过之前下载的所有视频。请注意，只有成功的下载才会记录在文件中。

例如，起初，

```
youtube-dl --download-archive archive.txt
"https://www.youtube.com/playlist?list=PLwiyx1dc3P2JR9N8gQaQN_BCvlSlap7re"
```

将下载完整的 `PLwiyx1dc3P2JR9N8gQaQN_BCvlSlap7re` 播放列表并创建一个文件 `archive.txt` 。每次后续运行只会下载新视频（如果有）：

```
youtube-dl --download-archive archive.txt
"https://www.youtube.com/playlist?list=PLwiyx1dc3P2JR9N8gQaQN_BCvlSlap7re"
```

我应该添加 `--hls-prefer-native` 到我的配置中吗?

当 youtube-dl 检测到 HLS 视频时，它可以使用内置下载器或 ffmpeg 下载它。由于许多 HLS 流有点无效，而且 ffmpeg/youtube-dl 都比另一个更好地处理一些无效情况，因此如果需要，可以选择切换下载器。

当 youtube-dl 知道某个特定的下载器对给定网站效果更好时，将选择该下载器。否则，youtube-dl 将选择最佳下载器以实现一般兼容性，目前恰好是 ffmpeg。随着内置下载器和/或 ffmpeg 的改进，此选择可能会在 youtube-dl 的未来版本中发生变化。

特别是，通用提取器（当您的网站不在youtube-dl 支持的网站列表中时使用）不能强制使用特定的下载器。

如果您将其中一个 `--hls-prefer-native` 或 `--hls-prefer-ffmpeg` 放入您的配置中，则不同的视频子集将无法正确下载。相反，最好提交问题或拉取请求，详细说明为什么本机或 ffmpeg HLS 下载器是您的用例的更好选择。

您可以添加对这个动漫视频网站或免费显示当前电影的网站的支持吗?

作为一项政策（以及合法性），youtube-dl 不包括对专门从事侵犯版权的服务的支持。根据经验，如果您无法轻易找到明显允许该服务分发的视频（即由创作者、创作者的分发者上传或根据免费许可发布的视频），则该服务可能不适合包含到 youtube-dl。

关于服务的说明，即他们不托管侵权内容，而只是链接到那些托管的内容，这表明该服务不应**包含**在 youtube-dl 中。当服务的整个首页充满了不允许分发的视频时，任何 DMCA 注释也是如此。如果服务在未经授权的情况下完整显示受版权保护的视频，则“合理使用”说明同样无法令人信服。

不过，对于购买了分发其内容的权利的服务的支持请求是完全可以的。如有疑问，您可以简单地包含提及合法购买内容的来源。

如何加快解决我的问题的速度?

(Also known as: Help, my important issue not being solved!) The youtube-dl core developer team is quite small. While we do our best to solve as many issues as possible, sometimes that can take quite a while. To speed up your issue, here's what you can do:

First of all, please do report the issue at our issue tracker. That allows us to coordinate all efforts by users and developers, and serves as a unified point. Unfortunately, the youtube-dl project has grown too large to use personal email as an effective communication channel.

Please read the bug reporting instructions below. A lot of bugs lack all the necessary information. If you can, offer proxy, VPN, or shell access to the youtube-dl developers. If you are able to, test the issue from multiple computers in multiple countries to exclude local censorship or misconfiguration issues.

If nobody is interested in solving your issue, you are welcome to take matters into your own hands and submit a pull request (or coerce/pay somebody else to do so).

Feel free to bump the issue from time to time by writing a small comment ("Issue is still present in youtube-dl version ...from France, but fixed from Belgium"), but please not more than once a month. Please do not declare your issue as `important` or `urgent`.

How can I detect whether a given URL is supported by youtube-dl?

For one, have a look at the list of supported sites. Note that it can sometimes happen that the site changes its URL scheme (say, from https://example.com/video/1234567 to https://example.com/v/1234567) and youtube-dl reports an URL of a service in that list as unsupported. In that case, simply report a bug.

*无法*检测 URL 是否受支持。这是因为 youtube-dl 包含一个匹配**所有**URL 的通用提取器。您可能想禁用、排除或删除通用提取器，但通用提取器不仅允许用户从许多嵌入其他服务的视频的网站中提取视频，还可以用于从服务中提取视频它自己托管。因此，我们既不推荐也不支持禁用、排除或删除通用提取器。

如果您想了解是否支持给定的 URL，只需调用 youtube-dl 即可。如果您没有收到任何视频，则该 URL 可能不是指视频或不受支持。您可以通过检查输出（如果您在控制台上运行 youtube-dl）或 `UnsupportedError` 从 Python 程序运行它时捕获异常来找出哪个。

为什么我在提交错误时需要经历那么多繁文缛节?

在我们有问题模板之前，尽管我们有大量的错误报告说明，但我们得到的问题报告中大约 80% 是无用的，例如因为人们使用了数百个版本的古老版本，因为简单的语法错误（不是在 youtube-dl 中，而是在一般的shell使用中），因为这个问题之前已经被多次报告过，因为人们实际上并没有阅读错误消息，即使它说“请安装ffmpeg”，因为人们没有提到他们试图下载的URL和许多更简单、易于避免的问题，其中许多与 youtube-dl 完全无关。

youtube-dl 是一个由太少志愿者管理的开源项目，所以我们宁愿花时间修复那些我们确定这些简单问题都不适用的错误，并且我们有足够的信心能够在没有问题的情况下重现问题记者反复询问。因此， 的输出 `youtube-dl -v YOUR_URL_HERE` 实际上是提交问题所需的全部内容。问题模板还指导您完成一些可以执行的基本步骤，例如检查您的 youtube-dl 版本是否为最新版本。

开发商说明

大多数用户不需要构建 youtube-dl，并且可以下载构建或从其分发中获取它们。

要以开发人员身份运行 youtube-dl，您也不需要构建任何东西。只需执行

```
python -m youtube_dl
```

要运行测试，只需调用您喜欢的测试运行器，或直接执行测试文件；以下任何工作：

```
python -m unittest discover
python test/test_download.py
nosetests
```

有关如何运行提取器特定测试用例的信息，请参阅新提取器教程的第 6 项。

如果你想自己创建一个 youtube-dl，你需要

- Python
- make（仅支持 GNU make）
- 潘多克
- 压缩
- 鼻子测试

添加对新站点的支持

如果您想添加对新站点的支持，首先**要确保**该站点**不是专门用于侵犯版权**的。youtube-dl 不**支持**此类站点，因此添加对它们的支持的拉取请求**将被拒绝**。

After you have ensured this site is distributing its content legally, you can follow this quick list (assuming your service is called `yourextractor`):

1. Fork this repository
2. Check out the source code with:

    ```
    git clone git@github.com:YOUR_GITHUB_USERNAME/youtube-dl.git
    ```

3. Start a new git branch with

```
cd youtube-dl
git checkout -b yourextractor
```

4. Start with this simple template and save it to `youtube_dl/extractor/yourextractor.py`:

```
# coding: utf-8
from __future__ import unicode_literals

from .common import InfoExtractor


class YourExtractorIE(InfoExtractor):
    _VALID_URL = r'https?://(?:www\.)?yourextractor\.com/watch/(?P<id>[0-9
    _TEST = {
        'url': 'https://yourextractor.com/watch/42',
        'md5': 'TODO: md5 sum of the first 10241 bytes of the video file (
        'info_dict': {
            'id': '42',
            'ext': 'mp4',
            'title': 'Video title goes here',
            'thumbnail': r're:^https?://.*\.jpg$',
            # TODO more properties, either as:
            # * A value
            # * MD5 checksum; start the string with md5:
            # * A regular expression; start the string with re:
            # * Any Python type (for example int or float)
        }
    }

    def _real_extract(self, url):
        video_id = self._match_id(url)
        webpage = self._download_webpage(url, video_id)

        # TODO more code goes here, for example ...
        title = self._html_search_regex(r'<h1>(.+?)</h1>', webpage, 'title

        return {
            'id': video_id,
            'title': title,
            'description': self._og_search_description(webpage),
            'uploader': self._search_regex(r'<div[^>]+id="uploader"[^>]*>(
            # TODO more properties (see youtube_dl/extractor/common.py)
        }
```

5. Add an import in `youtube_dl/extractor/extractors.py`.
6. Run `python test/test_download.py TestDownload.test_YourExtractor`. This *should fail* at first, but you can continually re-run it until you're done. If you decide to add more than one test, then rename `_TEST` to `_TESTS` and make it into a list of dictionaries. The tests will then be named `TestDownload.test_YourExtractor`, `TestDownload.test_YourExtractor_1`, `TestDownload.test_YourExtractor_2`, etc. Note that tests with `only_matching` key in test's dict are not counted in.
7. Have a look at `youtube_dl/extractor/common.py` for possible helper methods and a detailed description of what your extractor should and may return. Add tests and code for as many as you want.
8. Make sure your code follows youtube-dl coding conventions and check the code with flake8:

```
$ flake8 youtube_dl/extractor/yourextractor.py
```

9. Make sure your code works under all Python versions claimed supported by youtube-dl, namely 2.6, 2.7, and 3.2+.
10. When the tests pass, add the new files and commit them and push the result, like this:

```
$ git add youtube_dl/extractor/extractors.py
$ git add youtube_dl/extractor/yourextractor.py
$ git commit -m '[yourextractor] Add new extractor'
$ git push origin yourextractor
```

11. Finally, create a pull request. We'll then review and merge it.

In any case, thank you very much for your contributions!

youtube-dl coding conventions

This section introduces a guide lines for writing idiomatic, robust and future-proof extractor code.

Extractors are very fragile by nature since they depend on the layout of the source data provided by 3rd party media hosters out of your control and this layout tends to change. As an extractor implementer your task is not only to write code that will extract media links and metadata correctly but also to minimize dependency on the source's layout and even to make the code foresee potential future changes and be ready for that. This is important because it will allow the extractor not to break on minor layout changes thus keeping old youtube-dl versions working. Even though this breakage issue is easily fixed by emitting a new version of youtube-dl with a fix incorporated, all the previous versions become broken in all repositories and distros' packages that may not be so prompt in fetching the update from us. Needless to say, some non rolling release distros may never receive an update at all.

Mandatory and optional metafields

For extraction to work youtube-dl relies on metadata your extractor extracts and provides to youtube-dl expressed by an information dictionary or simply *info dict*. Only the following meta fields in the *info dict* are considered mandatory for a successful extraction process by youtube-dl:

- `id` (media identifier)
- `title` (media title)
- `url` (media download URL) or `formats`

In fact only the last option is technically mandatory (i.e. if you can't figure out the download location of the media the extraction does not make any sense). But by convention youtube-

自述文件.md

be extracted then the extractor is considered completely broken.

Any field apart from the aforementioned ones are considered **optional**. That means that extraction should be **tolerant** to situations when sources for these fields can potentially be unavailable (even if they are always available at the moment) and **future-proof** in order not to break the extraction of general purpose mandatory fields.

Example

Say you have some source dictionary `meta` that you've fetched as JSON with HTTP request and it has a key `summary`:

```
meta = self._download_json(url, video_id)
```

Assume at this point `meta`'s layout is:

```
{
    ...
    "summary": "some fancy summary text",
    ...
}
```

Assume you want to extract `summary` and put it into the resulting info dict as `description`. Since `description` is an optional meta field you should be ready that this key may be missing from the `meta` dict, so that you should extract it like:

```
description = meta.get('summary')  # correct
```

and not like:

```
description = meta['summary']  # incorrect
```

The latter will break extraction process with `KeyError` if `summary` disappears from `meta` at some later time but with the former approach extraction will just go ahead with `description` set to `None` which is perfectly fine (remember `None` is equivalent to the absence of data).

Similarly, you should pass `fatal=False` when extracting optional data from a webpage with `_search_regex`, `_html_search_regex` or similar methods, for instance:

```
description = self._search_regex(
    r'<span[^>]+id="title"[^>]*>([^<]+)<',
    webpage, 'description', fatal=False)
```

With `fatal` set to `False` if `_search_regex` fails to extract `description` it will emit a warning and continue extraction.

You can also pass `default=<some fallback value>`, for example:

```
description = self._search_regex(
    r'<span[^>]+id="title"[^>]*>([^<]+)<',
    webpage, 'description', default=None)
```

On failure this code will silently continue the extraction with `description` set to `None`. That is useful for metafields that may or may not be present.

Provide fallbacks

提取元数据时，请尝试从多个来源中提取。例如，如果 `title` 存在于多个地方，请尝试至少从其中一些地方提取。这使得它在某些来源变得不可用的情况下更具前瞻性。

例子

`meta` 从前面的例子中说有一个 `title` ，你将要提取它。因为 `title` 是一个强制性的元字段，你应该最终得到类似的东西：

```
标题 = 元[ '标题' ]
```

如果由于托管方方面的某些更改而 `title` 从将来消失，则提取将失败，因为这是强制性的。这是预期的。 `meta title`

假设您有一些可以从中提取的其他来源， `title` 例如. 在这种情况下，您可以提供一个备用方案： `og:title webpage`

```
标题 = 元。get ( 'title' )或 self。_og_search_title（网页）
```

此代码将尝试从 `meta` 第一个提取，如果失败，它将尝试 `og:title` 从 `webpage` .

常用表达

不要捕获您不使用的组

捕获组必须表明它在代码中的某处使用。任何未使用的组都必须是非捕获的。

例子

不要在此处捕获 id 属性名称，因为无论如何您都不能将其用于任何事情。

正确的：

```
r'(?:id|ID)=(?P<id>\d+)'
```

不正确：

```
r'(id|ID)=(?P<id>\d+)'
```

让正则表达式轻松灵活

使用正则表达式时，尝试将它们写得模糊、轻松和灵活，跳过更可能更改的无关紧要的部分，允许对引用的值使用单引号和双引号等。

例子

假设您需要 `title` 从以下 HTML 代码中提取：

```
< span  style =" position: absolute;left:910px;width:90px;float:right;z-index:
```

该任务的代码应类似于：

```
标题 = 自我。_search_regex （
     r'<span[^>]+class="title"[^>]*>([^<]+)' ,网页, 'title' ）
```

甚至更好：

```
标题 = 自我。_search_regex （
     r'<span[^>]+class=(["\'])title\1[^>]*>(?P<title>[^<]+)' ,
    网页, 'title' ,组= '标题'）
```

请注意您如何容忍属性值的潜在变化 `style` 或从使用双引号切换到单引号 `class` ：

代码绝对不应该是这样的：

```
标题 = 自我。_search_regex （
     r'<span style="position: absolute; left: 910px; width: 90px; float: right
    网页, '标题' ,组= '标题' ）
```

长线政策

将代码行保持在 80 个字符以下有一个软限制。这意味着如果可能并且它不会使可读性和代码维护变得更糟，则应该尊重它。

例如，您不应该**将**长字符串文字（如 URL 或其他一些经常复制的实体）拆分为多行以适应此限制：

正确的：

```
'https://www.youtube.com/watch?v=FqZTN594JQw&list=PLMYEtVRpaqY00V9W81Cwmzp6N6v
```

不正确：

```
'https://www.youtube.com/watch?v=FqZTN594JQw&list='
'PLMYEtVRpaqY00V9W81Cwmzp6N6vZqfUKD4'
```

内联值

提取变量对于减少代码重复和提高复杂表达式的可读性是可以接受的。但是，您应该避免提取仅使用一次的变量并将它们移动到提取器文件的相反部分，这会使读取线性流变得困难。

例子

正确的：

```
标题 = 自我。_html_search_regex ( r'<title>([^<]+)</title>' ,网页, 'title' )
```

不正确：

```
TITLE_RE  =  r'<title>([^<]+)</title>'
# ...一些代码行...
title  =  self . _html_search_regex ( TITLE_RE ,网页, 'title' )
```

折叠后备

多个后备值很快就会变得笨拙。通过模式列表将多个后备值折叠到一个表达式中。

例子

好的：

```
描述 = 自我。_html_search_meta (
    [ 'og:description' , 'description' , 'twitter:description' ],
    网页, 'description' ,默认= None )
```

笨重：

```
description = (
    self._og_search_description(webpage, default=None)
    or self._html_search_meta('description', webpage, default=None)
    or self._html_search_meta('twitter:description', webpage, default=None))
```

Methods supporting list of patterns are: `_search_regex` , `_html_search_regex` , `_og_search_property` , `_html_search_meta` .

Trailing parentheses

Always move trailing parentheses after the last argument.

Example

Correct:

```
    lambda x: x['ResultSet']['Result'][0]['VideoUrlSet']['VideoUrl'],
    list)
```

Incorrect:

```
    lambda x: x['ResultSet']['Result'][0]['VideoUrlSet']['VideoUrl'],
    list,
)
```

Use convenience conversion and parsing functions

Wrap all extracted numeric data into safe functions from `youtube_dl/utils.py`: `int_or_none`, `float_or_none`. Use them for string to number conversions as well.

Use `url_or_none` for safe URL processing.

Use `try_get` for safe metadata extraction from parsed JSON.

Use `unified_strdate` for uniform `upload_date` or any `YYYYMMDD` meta field extraction, `unified_timestamp` for uniform `timestamp` extraction, `parse_filesize` for `filesize` extraction, `parse_count` for count meta fields extraction, `parse_resolution`, `parse_duration` for `duration` extraction, `parse_age_limit` for `age_limit` extraction.

Explore `youtube_dl/utils.py` for more useful convenience functions.

More examples

Safely extract optional description from parsed JSON

```
description = try_get(response, lambda x: x['result']['video'][0]['summary'],
```

Safely extract more optional metadata

```
video = try_get(response, lambda x: x['result']['video'][0], dict) or {}
description = video.get('summary')
duration = float_or_none(video.get('durationMs'), scale=1000)
view_count = int_or_none(video.get('views'))
```

EMBEDDING YOUTUBE-DL

youtube-dl makes the best effort to be a good command-line program, and thus should be callable from any programming language. If you encounter any problems parsing its output, feel free to create a report.

From a Python program, you can embed youtube-dl in a more powerful fashion, like this:

```
from __future__ import unicode_literals
import youtube_dl

ydl_opts = {}
with youtube_dl.YoutubeDL(ydl_opts) as ydl:
    ydl.download(['https://www.youtube.com/watch?v=BaW_jenozKc'])
```

Most likely, you'll want to use various options. For a list of options available, have a look at `youtube_dl/YoutubeDL.py`. For a start, if you want to intercept youtube-dl's output, set a `logger` object.

Here's a more complete example of a program that outputs only errors (and a short message after the download is finished), and downloads/converts the video to an mp3 file:

```
from __future__ import unicode_literals
import youtube_dl


class MyLogger(object):
    def debug(self, msg):
        pass

    def warning(self, msg):
        pass

    def error(self, msg):
        print(msg)


def my_hook(d):
    if d['status'] == 'finished':
        print('Done downloading, now converting ...')


ydl_opts = {
    'format': 'bestaudio/best',
    'postprocessors': [{
        'key': 'FFmpegExtractAudio',
        'preferredcodec': 'mp3',
        'preferredquality': '192',
    }],
    'logger': MyLogger(),
```

```
    'progress_hooks': [my_hook],
}
with youtube_dl.YoutubeDL(ydl_opts) as ydl:
    ydl.download(['https://www.youtube.com/watch?v=BaW_jenozKc'])
```

BUGS

Bugs and suggestions should be reported at: https://github.com/ytdl-org/youtube-dl/issues. Unless you were prompted to or there is another pertinent reason (e.g. GitHub fails to accept the bug report), please do not send bug reports via personal email. For discussions, join us in the IRC channel #youtube-dl on freenode (webchat).

Please include the full output of youtube-dl when run with `-v`, i.e. **add** `-v` flag to **your command line**, copy the **whole** output and post it in the issue body wrapped in ``` for better formatting. It should look similar to this:

```
$ youtube-dl -v <your command line>
[debug] System config: []
[debug] User config: []
[debug] Command-line args: [u'-v', u'https://www.youtube.com/watch?
v=BaW_jenozKcj']
[debug] Encodings: locale cp1251, fs mbcs, out cp866, pref cp1251
[debug] youtube-dl version 2015.12.06
[debug] Git HEAD: 135392e
[debug] Python version 2.6.6 - Windows-2003Server-5.2.3790-SP2
[debug] exe versions: ffmpeg N-75573-g1d0487f, ffprobe N-75573-g1d0487f,
rtmpdump 2.4
[debug] Proxy map: {}
...
```

Do not post screenshots of verbose logs; only plain text is acceptable.

The output (including the first lines) contains important debugging information. Issues without the full output are often not reproducible and therefore do not get solved in short order, if ever.

Please re-read your issue once again to avoid a couple of common mistakes (you can and should use this as a checklist):

Is the description of the issue itself sufficient?

We often get issue reports that we cannot really decipher. While in most cases we eventually get the required information after asking back multiple times, this poses an unnecessary drain on our resources. Many contributors, including myself, are also not native speakers, so we may misread some parts.

So please elaborate on what feature you are requesting, or what bug you want to be fixed. Make sure that it's obvious

- What the problem is
- How it could be fixed
- How your proposed solution would look like

If your report is shorter than two lines, it is almost certainly missing some of these, which makes it hard for us to respond to it. We're often too polite to close the issue outright, but the missing info makes misinterpretation likely. As a committer myself, I often get frustrated by these issues, since the only possible way for me to move forward on them is to ask for clarification over and over.

For bug reports, this means that your report should contain the *complete* output of youtube-dl when called with the `-v` flag. The error message you get for (most) bugs even says so, but you would not believe how many of our bug reports do not contain this information.

If your server has multiple IPs or you suspect censorship, adding `--call-home` may be a good idea to get more diagnostics. If the error is `ERROR: Unable to extract ...` and you cannot reproduce it from multiple countries, add `--dump-pages` (warning: this will yield a rather large output, redirect it to the file `log.txt` by adding `>log.txt 2>&1` to your command-line) or upload the `.dump` files you get when you add `--write-pages` somewhere.

Site support requests must contain an example URL. An example URL is a URL you might want to download, like `https://www.youtube.com/watch?v=BaW_jenozKc` . There should be an obvious video present. Except under very special circumstances, the main page of a video service (e.g. `https://www.youtube.com/`) is *not* an example URL.

Are you using the latest version?

Before reporting any issue, type `youtube-dl -U` . This should report that you're up-to-date. About 20% of the reports we receive are already fixed, but people are using outdated versions. This goes for feature requests as well.

Is the issue already documented?

Make sure that someone has not already opened the issue you're trying to open. Search at the top of the window or browse the GitHub Issues of this repository. If there is an issue, feel free to write something along the lines of "This affects me as well, with version 2015.01.01. Here is some more information on the issue: ...". While some issues may be old, a new post into them often spurs rapid activity.

Why are existing options not enough?

Before requesting a new feature, please have a quick peek at the list of supported options. Many feature requests are for features that actually exist already! Please, absolutely do show off your work in the issue report and detail how the existing similar options do *not* solve your problem.

Is there enough context in your bug report?

People want to solve problems, and often think they do us a favor by breaking down their larger problems (e.g. wanting to skip already downloaded files) to a specific request (e.g. requesting us to look whether the file exists before downloading the info page). However, what often happens is that they break down the problem into two steps: One simple, and one impossible (or extremely complicated one).

We are then presented with a very complicated request when the original problem could be solved far easier, e.g. by recording the downloaded video IDs in a separate file. To avoid this, you must include the greater context where it is non-obvious. In particular, every feature request that does not consist of adding support for a new site should contain a use case scenario that explains in what situation the missing feature would be useful.

Does the issue involve one problem, and one problem only?

Some of our users seem to think there is a limit of issues they can or should open. There is no limit of issues they can or should open. While it may seem appealing to be able to dump all your issues into one ticket, that means that someone who solves one of your issues cannot mark the issue as closed. Typically, reporting a bunch of issues leads to the ticket lingering since nobody wants to attack that behemoth, until someone mercifully splits the issue into multiple ones.

In particular, every site support request issue should only pertain to services at one site (generally under a common domain, but always using the same backend technology). Do not request support for vimeo user videos, White house podcasts, and Google Plus pages in the same issue. Also, make sure that you don't post bug reports alongside feature requests. As a rule of thumb, a feature request does not include outputs of youtube-dl that are not immediately related to the feature at hand. Do not post reports of a network error alongside the request for a new video service.

Is anyone going to need the feature?

Only post features that you (or an incapacitated friend you can personally talk to) require. Do not post features because they seem like a good idea. If they are really useful, they will be requested by someone who requires them.

你的问题是关于 youtube-dl 的吗?

听起来可能很奇怪，但我们收到的一些错误报告与 youtube-dl 完全无关，并且与不同的应用程序有关，甚至与报告者自己的应用程序有关。请确保您实际使用的是 youtube-dl。如果您使用 youtube-dl 的 UI，请将错误报告给提供 UI 的实际应用程序的维护者。另一方面，如果您的 youtube-dl 用户界面以某种方式失败，您认为这与 youtube-dl 相关，请务必报告该错误。

版权

youtube-dl 由版权所有者发布到公共领域。

此 README 文件最初由Daniel Bolton编写，同样已发布到公共领域。

发布 341

youtube-dl 2021.06.06 最新的
on 6 Jun 2021

+ 340 个版本

套餐

没有发布包

贡献者 768

+ 757 位贡献者

语言

Python 99.6% **其他** 0.4%