
소용석, 김예찬, 이성필 팀
이정현 : scripts/pytube 분석

scripts

pytube

:: 파싱을 수행하기 위한 파서 선언 및 command-line 인터페이스 형식 정의
(다운로드 대상 비디오 1개 추출 및 다운로드 할 수 있도록 정보를 전달하는 역할)

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import print_function
import sys
import os
import argparse

from pytube import YouTube
from pytube.utils import print_status, FullPaths
from pytube.exceptions import PytubeError
from pprint import pprint


def main():
    parser = argparse.ArgumentParser(description='YouTube video downloader')
    parser.add_argument("url", help=(
        "The URL of the Video to be downloaded"))
    parser.add_argument("--extension", "-e", dest="ext", help=(
        "The requested format of the video"))
    parser.add_argument("--resolution", "-r", dest="res", help=(
        "The requested resolution"))
    parser.add_argument("--path", "-p", action=FullPaths, default=os.getcwd(),
                        dest="path", help=("The path to save the video to."))
    parser.add_argument("--filename", "-f", dest="filename", help=(
        "The filename, without extension, to save the video in."))

    args = parser.parse_args()

    try:
        yt = YouTube(args.url)
        videos = []
        for i, video in enumerate(yt.get_videos()):
            ext = video.extension
            res = video.resolution
            videos.append("{} {}".format(ext, res))
    except PytubeError:
        print("Incorrect video URL.")
        sys.exit(1)

    if args.filename:
        yt.set_filename(args.filename)

    if args.ext or args.res:
        if not all([args.ext, args.res]):
            print("Make sure you give either of the below specified "
                  "format/resolution combination.")
            pprint(videos)
            sys.exit(1)

    if args.ext and args.res:
        # There's only ope video that matches both so get it
        vid = yt.get(args.ext, args.res)
        # Check if there's a video returned
        if not vid:
            print("There's no video with the specified format/resolution "
                  "combination.")
            pprint(videos)
            sys.exit(1)

    elif args.ext:
        # There are several videos with the same extension
        videos = yt.filter(extension=args.ext)
        # Check if we have a video
        if not videos:
            print("There are no videos in the specified format.")
            sys.exit(1)
        # Select the highest resolution one
        vid = max(videos)
    elif args.res:
        # There might be several videos in the same resolution
        videos = yt.filter(resolution=args.res)
        # Check if we have a video
        if not videos:
            print("There are no videos in the specified in the specified "
                  "resolution.")
            sys.exit(1)
        # Select the highest resolution one
        vid = max(videos)
    else:
        # If nothing is specified get the highest resolution one
        vid = max(yt.get_videos())

    try:
        vid.download(path=args.path, on_progress=print_status)
    except KeyboardInterrupt:
        print("Download interrupted.")
        sys.exit(1)

if __name__ == '__main__':
    main()
```

## if __name__ == "__main__"

## '만일 이 파일이 interpreter에 의해서 실행되는 경우라면' 의 의미

구현한 코드가가 다른 파이썬 코드에 의해 module로 import 될 경우도 있고, python interpreter에 의해 직접 실행될 경우도 있는데 위 코드는 interpreter에 의해 직접 실행 될 경우에만 실행하도록 하고 싶은 코드 블록이 있을 경우 사용

[참고] http://stackoverflow.com/questions/419163/what-does-if-name-main-do

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import print_function
import sys
import os
import argparse

from pytube import YouTube
from pytube.utils import print_status, FullPaths
from pytube.exceptions import PytubeError
from pprint import pprint


def main():
    parser = argparse.ArgumentParser(description='YouTube video downloader')
    parser.add_argument("url", help=(
        "The URL of the Video to be downloaded"))
    parser.add_argument("--extension", "-e", dest="ext", help=(
        "The requested format of the video"))
    parser.add_argument("--resolution", "-r", dest="res", help=(
        "The requested resolution"))
    parser.add_argument("--path", "-p", action=FullPaths, default=os.getcwd(),
                        dest="path", help=("The path to save the video to."))
    parser.add_argument("--filename", "-f", dest="filename", help=(
        "The filename, without extension, to save the video in."))

    args = parser.parse_args()
```

```python
    try:
        yt = YouTube(args.url)
        videos = []
        for i, video in enumerate(yt.get_videos()):
            ext = video.extension
            res = video.resolution
            videos.append("{} {}".format(ext, res))
    except PytubeError:
        print("Incorrect video URL.")
        sys.exit(1)

    if args.filename:
        yt.set_filename(args.filename)

    if args.ext or args.res:
        if not all([args.ext, args.res]):
            print("Make sure you give either of the below specified "
                  "format/resolution combination.")
            pprint(videos)
            sys.exit(1)

    if args.ext and args.res:
        # There's only ope video that matches both so get it
        vid = yt.get(args.ext, args.res)
        # Check if there's a video returned
        if not vid:
            print("There's no video with the specified format/resolution "
                  "combination.")
            pprint(videos)
```

**`import argparse`** — **Argparse 모듈**

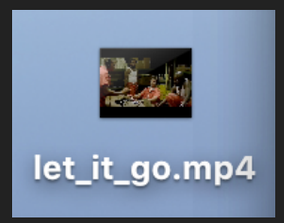
명령행 인터페이스(커맨드 라인 인터페이스)에서 명령행 인자를 받아서 실행되는 프로그램의 경우
**주어진 인자들을 파싱하는 작업이 필요하여 사용되는 모듈**

[참고] https://docs.python.org/2/library/argparse.html#argparse.ArgumentParser

**PYTUBE를 아래와 같은 형식으로 사용하기 위하여…**

```
>>> from pytube import YouTube
>>> yt = YouTube("https://youtu.be/4-cAjYzQncQ")
>>> video = yt.get('mp4','720p')
>>> yt.set_filename('let_it_go')
True
>>> video.download('/Users/sodarain730/Desktop')
```

console(command-line interface)에 명령 입력




/Users/sodarain730/Desktop경로에 저장된 결과물

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import print_function
import sys
import os
import argparse

from pytube import YouTube
from pytube.utils import print_status, FullPaths
from pytube.exceptions import PytubeError
from pprint import pprint


def main():
    parser = argparse.ArgumentParser(description='YouTube video downloader')
    parser.add_argument("url", help=(
        "The URL of the Video to be downloaded"))
    parser.add_argument("--extension", "-e", dest="ext", help=(
        "The requested format of the video"))
    parser.add_argument("--resolution", "-r", dest="res", help=(
        "The requested resolution"))
    parser.add_argument("--path", "-p", action=FullPaths, default=os.getcwd(),
                        dest="path", help=("The path to save the video to."))
    parser.add_argument("--filename", "-f", dest="filename", help=(
        "The filename, without extension, to save the video in."))

    args = parser.parse_args()

    try:
        yt = YouTube(args.url)
        videos = []
        for i, video in enumerate(yt.get_videos()):
            ext = video.extension
            res = video.resolution
            videos.append("{} {}".format(ext, res))
    except PytubeError:
        print("Incorrect video URL.")
        sys.exit(1)

    if args.filename:
        yt.set_filename(args.filename)

    if args.ext or args.res:
        if not all([args.ext, args.res]):
            print("Make sure you give either of the below specified "
                  "format/resolution combination.")
            pprint(videos)
            sys.exit(1)

    if args.ext and args.res:
        # There's only ope video that matches both so get it
        vid = yt.get(args.ext, args.res)
        # Check if there's a video returned
        if not vid:
            print("There's no video with the specified format/resolution "
                  "combination.")
            pprint(videos)
            sys.exit(1)

    elif args.ext:
        # There are several videos with the same extension
        videos = yt.filter(extension=args.ext)
        # Check if we have a video
        if not videos:
            print("There are no videos in the specified format.")
            sys.exit(1)
        # Select the highest resolution one
        vid = max(videos)
    elif args.res:
        # There might be several videos in the same resolution
        videos = yt.filter(resolution=args.res)
        # Check if we have a video
        if not videos:
            print("There are no videos in the specified in the specified "
                  "resolution.")
            sys.exit(1)
        # Select the highest resolution one
        vid = max(videos)
    else:
        # If nothing is specified get the highest resolution one
        vid = max(yt.get_videos())

    try:
        vid.download(path=args.path, on_progress=print_status)
    except KeyboardInterrupt:
        print("Download interrupted.")
        sys.exit(1)

if __name__ == '__main__':
    main()
```

파싱을 수행하기 위한 **파서 선언**

## 파싱[parsing]

컴퓨터에서 컴파일러 또는 번역기가 원시 부호를 기계어로 번역하는 과정의 한 단계로, 각 문장의 문법적인 구성 또는 구문을 분석하는 과정. 즉 원시 프로그램에서 나타난 토큰(token)의 열을 받아들여 이를 그 언어의 문법에 맞게 구문 분석 트리(parse tree)로 구성해 내는 일이다. 자료를 원하는 형태로 가공하기 위한 것

## 파서[parser]

컴퓨팅에서 파서(parser)는 인터프리터나 컴파일러의 구성 요소 가운데 하나로, 입력 토큰에 내재된 자료 구조를 빌드하고 문법을 검사한다. 파서는 일련의 입력 문자로부터 토큰을 만들기 위해 별도의 낱말 분석기를 이용하기도 한다. 파서는 수작업으로 프로그래밍되며 도구에 의해 (일부 프로그래밍 언어에서) (반)자동적으로 만들어질 수 있다.

```
parser = argparse.ArgumentParser(description='YouTube video downloader')
```

argparse.**ArgumentParser** 클래스는 명령줄 옵션을 파싱하는 객체
Argpase를 사용하기 위한 첫 번째 단계는 parser 객체를 생성,

```
parser.add_argument("url", help=(
    "The URL of the Video to be downloaded"))
parser.add_argument("--extension", "-e", dest="ext", help=(
    "The requested format of the video"))
parser.add_argument("--resolution", "-r", dest="res", help=(
    "The requested resolution"))
parser.add_argument("--path", "-p", action=FullPaths, default=os.getcwd(),
                    dest="path", help=("The path to save the video to."))
parser.add_argument("--filename", "-f", dest="filename", help=(
    "The filename, without extension, to save the video in."))
```

객체 생성후에는 **add_argument()** 함수를 통해서 각 스위치와 그 옵션들을 추가해주고, 최종적으로 전달된 arguments들을 던져주면 사전형태(추가한 argument들을 대상으로 하는 네임스페이스를 구비한 객체를 전달한다. 왜냐하면 동일한 기능을 하는 스위치를 2개 이상의 이름을 줄 수 있기 때문이다. 예를 들어 -p와 --path는 동일한 스위치로 둘 수 있는데, 이 경우 args.p 나 args.path 둘 모두로 동일한 값에 접근가능하다.)로 이를 돌려주게 된다.

:: 파서객체의 **add_argument()** 메소드는 파라미터 파싱처리의 핵심을 담당

.add_argument(name or flags...[,action][,nargs][,default][,type][,choices][,required][,help][,metavar][,dest])

- name or flags : 등록할 파라미터의 이름이나 스위치를 등록한다. “foo”, “-f”, “–foo” 등이 가능하다.
- dest : 스위치나 파라미터이름이 아닌 별도의 변수를 지정할 때 쓴다. 외부에서 변수를 미리 선언한 경우, 해당 변수에 값이 들어간다.
- action: 스위치가 주어졌을 때, 표준 동작을 정한다.
- help: –help 옵션을 받았을 때, 표시될 메시지 목록에서 스위치의 도움말을 설정한다.

```
args = parser.parse_args()
```

**.parse_args()** :: 각 인자의 리스트를 받아 이를 파싱한 결과를 되돌려 준다.
파싱된 결과는 하나의 네임스페이스를 갖는 객체로 모든 스위치 이름들은 이 객체의 속성 이름으로 정의되어 있다. 만약 인자값을 넘기지 않는다면 기본적으로 sys.argv를 사용하게 된다.

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import print_function
import sys
import os
import argparse

from pytube import YouTube
from pytube.utils import print_status, FullPaths
from pytube.exceptions import PytubeError
from pprint import pprint


def main():
    parser = argparse.ArgumentParser(description='YouTube video downloader')
    parser.add_argument("url", help=(
        "The URL of the Video to be downloaded"))
    parser.add_argument("--extension", "-e", dest="ext", help=(
        "The requested format of the video"))
    parser.add_argument("--resolution", "-r", dest="res", help=(
        "The requested resolution"))
    parser.add_argument("--path", "-p", action=FullPaths, default=os.getcwd(),
                        dest="path", help=("The path to save the video to."))
    parser.add_argument("--filename", "-f", dest="filename", help=(
        "The filename, without extension, to save the video in."))

    args = parser.parse_args()

    try:
        yt = YouTube(args.url)
        videos = []
        for i, video in enumerate(yt.get_videos()):
            ext = video.extension
            res = video.resolution
            videos.append("{} {}".format(ext, res))
    except PytubeError:
        print("Incorrect video URL.")
        sys.exit(1)

    if args.filename:
        yt.set_filename(args.filename)

    if args.ext or args.res:
        if not all([args.ext, args.res]):
            print("Make sure you give either of the below specified "
                  "format/resolution combination.")
            pprint(videos)
            sys.exit(1)

    if args.ext and args.res:
        # There's only ope video that matches both so get it
        vid = yt.get(args.ext, args.res)
        # Check if there's a video returned
        if not vid:
            print("There's no video with the specified format/resolution "
                  "combination.")
            pprint(videos)
```

## 1) ArgumentParser class로 parser객체 생성(선언)

```python
class ArgumentParser(_AttributeHolder, _ActionsContainer):
    """Object for parsing command line strings into Python objects...."""

    def __init__(self,
                 prog=None,
                 usage=None,
                 description=None,
                 epilog=None,
                 parents=[],
                 formatter_class=HelpFormatter,
                 prefix_chars='-',
                 fromfile_prefix_chars=None,
                 argument_default=None,
                 conflict_handler='error',
                 add_help=True,
                 allow_abbrev=True):...

    # =======================
    # Pretty __repr__ methods
    # =======================
    def _get_kwargs(self):...
```

## 2) add_argument method로 parse 인수 추가

```python
# =======================
# Adding argument actions
# =======================
def add_argument(self, *args, **kwargs):
    """
    add_argument(dest, ..., name=value, ...)
    add_argument(option_string, option_string, ..., name=value, ...)
    """
```

args 객체는 명령행을 수행하는 데 필요한 모든 정보를 담게 된다

## 3) parse_args()를 통해 ArgmentParser 인수 분석

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import print_function
import sys
import os
import argparse

from pytube import YouTube
from pytube.utils import print_status, FullPaths
from pytube.exceptions import PytubeError
from pprint import pprint


def main():
    parser = argparse.ArgumentParser(description='YouTube video downloader')
    parser.add_argument("url", help=(
        "The URL of the Video to be downloaded"))
    parser.add_argument("--extension", "-e", dest="ext", help=(
        "The requested format of the video"))
    parser.add_argument("--resolution", "-r", dest="res", h
        "The requested resolution"))
    parser.add_argument("--path", "-p", action=FullPaths, d
                        dest="path", help=("The path to sav
    parser.add_argument("--filename", "-f", dest="filename"
        "The filename, without extension, to save the video

    args = parser.parse_args()

    try:
        yt = YouTube(args.url)
        videos = []
        for i, video in enumerate(yt.get_videos()):
            ext = video.extension
            res = video.resolution
            videos.append("{} {}".format(ext, res))
    except PytubeError:
        print("Incorrect video URL.")
        sys.exit(1)

    if args.filename:
        yt.set_filename(args.filename)

    if args.ext or args.res:
        if not all([args.ext, args.res]):
            print("Make sure you give either of the below specified "
                  "format/resolution combination.")
            pprint(videos)
            sys.exit(1)

    if args.ext and args.res:
        # There's only ope video th
        vid = yt.get(args.ext, args
        # Check if there's a video
        if not vid:
            print("There's no video
                  "combination.")
            pprint(videos)
            sys.exit(1)

    elif args.ext:
        # There are several videos with the same extension
        videos = yt.filter(extension=args.ext)
        # Check if we have a video
        if not videos:
            print("There are no videos in the specified format.")
            sys.exit(1)
        # Select the highest resolution one
        vid = max(videos)
    elif args.res:
        # There might be several videos in the same resolution
        videos = yt.filter(resolution=args.res)
        # Check if we have a video
        if not videos:
            print("There are no videos in the specified in the specified "
                  "format/resolution.")
            sys.exit(1)
        # Select the highest resolution one
        vid = max(videos)
    else:
        # No specified get the highest resolution one
        vid = max(yt.get_videos())

    try:
        vid.download(path=args.path, on_progress=print_status)
    except KeyboardInterrupt:
        print("Download interrupted.")
        sys.exit(1)
```

api.py

```python
def get_videos(self):
    """Gets all videos."""
    return self._videos
```

models.py

```python
class Video(object):
    """Class representation of a single instance of a YouTube video.
    """
    def __init__(self, url, filename, extension, resolution=None,
                 video_codec=None, profile=None, video_bitrate=None,
                 audio_codec=None, audio_bitrate=None):
        """Sets-up the video object.

        :param str extention:
            The desired file extention (e.g.: mp4, flv, webm).
        :param str resolution:
            *Optional* The broadcasting standard (e.g.: 720p, 1080p).

        self.extension = extension
        self.resolution = resolution
```

command-line 에 입력받은 url에 해당하는 비디오들(하나의 url을 입력 하면 하나의 비디오가 오는 것이 아니라 확장자, 해상도 별로 여러개 올 수 있음)을 Binary형태의 array인 videos로 만든다

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import print_function
import sys
import os
import argparse

from pytube import YouTube
from pytube.utils import print_status, FullPaths
from pytube.exceptions import PytubeError
from pprint import pprint


def main():
    parser = argparse.ArgumentParser(description='YouTube video downloader')
    parser.add_argument("url", help=(
        "The URL of the video to be downloaded"))
```

api.py

```python
def set_filename(self, filename):
    """Sets the filename of the video.

    :param str filename:
        The filename of the video.
    """
    # TODO: Check if the filename contains the file extension and either
    # strip it or raise an exception.
    self._filename = filename
    if self.get_videos():
        for video in self.get_videos():
            video.filename = filename
    return True
```

입력 받은 파일이름 적용

```python
    for i, video in enumerate(yt.get_videos(), 1):
        ext = video.extension
        res = video.resolution
        videos.append("{} {}".format(ext, res))
except PytubeError:
    print("Incorrect video URL.")
    sys.exit(1)

if args.filename:
    yt.set_filename(args.filename)

if args.ext or args.res:
    if not all([args.ext, args.res]):
        print("Make sure you give either of the below specified "
              "format/resolution combination.")
        pprint(videos)
        sys.exit(1)

if args.ext and args.res:
    # There's only ope video that matches both so get it
    vid = yt.get(args.ext, args.res)
    # Check if there's a video returned
    if not vid:
        print("There's no video with the specified format/resolution "
              "combination.")
        pprint(videos)
        sys.exit(1)

elif args.ext:
    # There are several videos with the same extension
    videos = yt.filter(extension=args.ext)
    # Check if we have a video
    if not videos:
        print("There are no videos in the specified format.")
        sys.exit(1)
    # Select the highest resolution one
    vid = max(videos)
elif args.res:
    # There might be several videos in the same resolution
    videos = yt.filter(resolution=args.res)
```

api.py

```python
def get(self, extension=None, resolution=None, profile=None):
    """Gets a single video given a file extention (and/or resolution
    and/or quality profile).

    :param str extention:
        The desired file extention (e.g.: mp4, flv).
    :param str resolution:
        The desired video broadcasting standard (e.g.: 720p, 1080p)
    :param str profile:
        The desired quality prof                              recommend
        using it).
    """
    result = []
    for v in self.get_videos():
        if extension and v.extension != extension:
            continue
        elif resolution and v.resolution != resolution:
            continue
        elif profile and v.profile != profile:
            continue
        else:
            result.append(v)
    matches = len(result)
    if matches <= 0:
        raise DoesNotExist("No videos met this criteria.")
    elif matches == 1:
        return result[0]
    else:
        raise MultipleObjectsReturned("Multiple videos met this criteria.")
```

```python
def get_videos(self):
    """Gets all videos."""
    return self._videos
```

if, self.get_videos() 를 통해 가져온 비디오들(해상도별,확장자별로 모두 가져옴)의 해상도와 확장자가 사용자가 입력한 정보와 모두 일치하는 비디오를 찾고 해당하는 비디오가 1개 일 경우에만 결과를 리턴하여 vid객체로 만든다

api.py

```python
def filter(self, extension=None, resolution=None, profile=None):
    """..."""
    results = []
    for v in self.get_videos():
        if extension and v.extension != extension:
            continue
        elif resolution and v.resolution != resolution:
            continue
        elif profile and v.profile != profile:
            continue
        else:
            results.append(v)
    return results
```

elif, 같은 확장자를 가진 여러 비디오가 있다면 그 중 가장 해상도가 높은 비디오 1개를 vid객체로 만든다.

elif, 같은 해상도를 가진 여러 비디오가 있다면 그 중 1개를 vid객체로 만든다.

else, 아무 것도 매칭되는 정보가 없다면 가장 높은 해상도 기준으로 1개의 비디오를 선택, vid객체를 만든다.

위의 과정을 통해 선택 된 1개의 비디오를 download 하기 위하여 models.py에 선택 된 비디오의 path와 on_progress 키워드 인자를 넘겨준다.

```python
    sys.exit(1)

elif args.ext:
    # There are several videos with the same extension
    videos = yt.filter(extension=args.ext)
    # Check if we have a video
    if not videos:
        print("There are no videos in the specified format.")
        sys.exit(1)
    # Select the highest resolution one
    vid = max(videos)
elif args.res:
    # There might be several videos in the same resolution
    videos = yt.filter(resolution=args.res)
    # Check if we have a video
    if not videos:
        print("There are no videos in the specified in the specified "
              "resolution.")
        sys.exit(1)
    # Select the highest resolution one
    vid = max(videos)
else:
    # If nothing is specified get the highest resolution one
    vid = max(yt.get_videos())

try:
    vid.download(path=args.path, on_progress=print_status)
except KeyboardInterrupt:
    print("Download interrupted.")
    sys.exit(1)
```

models.py

```python
def download(self, path, chunk_size=8 * 1024, on_progress=None,
             on_finish=None, force_overwrite=False):
    """Downloads the video...."""
    path = os.path.normpath(path)
    if os.path.isdir(path):
        filename = "{0}.{1}".format(self.filename, self.extension)
        path = os.path.join(path, filename)
    # TODO: If it's not a path, this should raise an ``OSError``.
    # TODO: Move this into cli, this kind of logic probably shouldn't be
    # handled by the library.
    if os.path.isfile(path) and not force_overwrite:
        raise OSError("Conflicting filename:'{0}'".format(self.filename))
    # TODO: Split up the downloading and OS jazz into separate functions.
    response = urlopen(self.url)
    meta_data = dict(response.info().items())
    file_size = int(meta_data.get("Content-Length") or
                    meta_data.get("content-length"))
    self._bytes_received = 0
    start = clock()
    # TODO: Let's get rid of this whole try/except block, let ``OSErrors``
    # fail loudly.
    try:
        with open(path, 'wb') as dst_file:
            while True:
                self._buffer = response.read(chunk_size)
                # Check if the buffer is empty (aka no bytes remaining).
                if not self._buffer:
                    if on_finish:
```

# 예외처리

```
pytube
.idea
pytube
    __init__.py
    api.py
    compat.py
    exceptions.py
    jsinterp.py
    models.py
    utils.py
scripts
    pytube
```

```
class PytubeError(Exception):
    """Something specific to the wrapper failed.
    """
    pass
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import print_function

from pytube import YouTube
from pytube.utils import print_status, FullPaths
from pytube.exceptions import PytubeError
from pprint import pprint

umentParser(description='YouTube video downloader')
url", help=(
ideo to be downloaded"))
--extension", "-e", dest="ext", help=(
rmat of the video"))
-resolution", "-r", dest="res", help=(
olution"))
),
)

args = parser.parse_args()

try:
    yt = YouTube(args.url)
    videos = []
    for i, video in enumerate(yt.get_videos()):
        ext = video.extension
        res = video.resolution
        videos.append("{} {}".format(ext, res))
except PytubeError:
    print("Incorrect video URL.")
    sys.exit(1)

if args.filename:
    yt.set_filename(args.filename)

if args.ext or args.res:
    if not all([args.ext, args.res]):
        print("Make sure you give either of the below specified "
              "format/resolution combination.")
        pprint(videos)
        sys.exit(1)

if args.ext and args.res:
    # There's only ope video that matches both so get it
    vid = yt.get(args.ext, args.res)
    # Check if there's a video returned
    if not vid:
        print("There's no video with the specified format/resolution "
              "combination.")
        pprint(videos)
        sys.exit(1)

elif args.ext:
    # There are several videos with the same extension
    videos = yt.filter(extension=args.ext)
    # Check if we have a video
    if not videos:
        print("There are no videos in the specified format.")
        sys.exit(1)
    # Select the highest resolution one
    vid = max(videos)
elif args.res:
    # There might be several videos in the same resolution
    videos = yt.filter(resolution=args.res)
    # Check if we have a video
    if not videos:
        print("There are no videos in the specified in the specified "
              "resolution.")
        sys.exit(1)
    # Select the highest resolution one
    vid = max(videos)
else:
    # If nothing is specified get the highest resolution one
    vid = max(yt.get_videos())

try:
    vid.download(path=args.path, on_progress=print_status)
except KeyboardInterrupt:
    print("Download interrupted.")
    sys.exit(1)

if __name__ == '__main__':
    main()
```

**sys.exit(1)**

exit(0)는 정상종료, exit(1)은 비 정상 종료로 간주