TUBTUB TEAM

PYTUBE 분석

TUBTUB TEAM

API.PY - 승권

분배

- 박승권: api.py
- 서은정: model.py
- 유동현: util.py
- 박선배: jsinterp.py

API.PY

```
class YouTube(object):
    """Class representation of a single instance of a YouTube session.
    """
    def __init__(self, url=None):...

    @property
    def url(self):...

    @url.setter
    def url(self, url):...

    @property
    def video_id(self):...

    @property
    def filename(self):...

    @filename.setter
    def filename(self, filename):...

    def set_filename(self, filename):...

    def get_videos(self):...

    @property
    def videos(self):...

    def from_url(self, url):...

    def get(self, extension=None, resolution=None, profile=None):...

    def filter(self, extension=None, resolution=None, profile=None):...

    # 비디오의 어떠한 데이터를 가져올까?
    def get_video_data(self):...

    def _parse_stream_map(self, blob):...

    def _get_json_data(self, html):...

    def _get_json_offset(self, html):...

    def _get_cipher(self, signature, url):...

    def _get_quality_profile_from_url(self, video_url):...

    def _add_video(self, url, filename, **kwargs):...
```

- getter, setter를 제외한 11개 함수 총 19개
- 동영상 형식이나 제한등에 따라 사용하지 않는 함수들도 있다.
- 수업시간에 배운 getter, setter의 의미를 명확히 이해할 수 있다.
- 새로운 알고리즘을 익힐 수 있다.

다운로드를 기준으로 작성

```python
from pytube import YouTube

# youtube("https://www.youtube.com/watch?v=22mdHYEzKC0","Paint")

yt = YouTube("https://www.youtube.com/watch?v=22mdHYEzKC0")
yt.set_filename("paint")
video = yt.get('mp4','720p')
video.download('/Users/yevgnenll/Desktop/')
```

- YouTube(“url”) 입력

```python
def __init__(self, url=None):  self: <pytube
    """Initializes YouTube API wrapper.

    :param str url:
        The url to the YouTube video.
    """
    self._filename = None
    self._video_url = None
    self._js_cache = None
    self._videos = []
    if url:
        self.from_url(url)
```

- 받은 url을 set으로 알려줌.
- from_url(url)로 하면

다운로드를 기준으로 작성

```
def from_url(self, url):
    """Sets the url for the video.

    :param str url:
        The url to the YouTube video.(파일의 주소를 셋팅)
    """
    self._video_url = url

    # Reset the filename and videos list in case the same instance is
    # reused.
    self._filename = None
    self._videos = []

    # Get the video details.
    video_data = self.get_video_data()

    # Set the title from the title.
    self.title = video_data.get("args", {}).get("title")
```

- from_url 함수로 들어가서 클래스 내부에 url을 저장해줌.
- filename을 재사용 되는 변수이기 때문에 무조건 None으로 초기화함.
- 비디오 데이터를 가져옴

```
# 비디오의 어떠한 데이터를 가져올까?
def get_video_data(self):  self: <pytube.api.YouTube object at 0x1034cf748>
    """Gets the page and extracts out the video data."""
    # Reset the filename incase it was previously set.
    self.title = None
    response = urlopen(self.url)
```

- get_video_data 에서 제목을 초기화 함.
- urlopen에선 http 통신을 시작함.

다운로드를 기준으로 작성

```
@property
def url(self):  self: <pytube.api.YouTube object at 0x1034cf748>
    """Gets the video url."""
    return self._video_url
```

- 맨 처음 저장한 url을 가져온다

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3   # flake8: noqa
4   import sys
5
6   PY2 = sys.version_info[0] == 2
7   PY3 = sys.version_info[0] == 3
8
9   if PY2:
10      from urllib2 import urlopen
11      from urlparse import urlparse, parse_qs, unquote
12  if PY3:
13      from urllib.parse import urlparse, parse_qs, unquote
14      from urllib.request import urlopen
15
```

- urlopen은 내장된 라이브러리이다
  - url로 요청을 보낼때 사용
  - 요청 이후 response를 받아옴.

다운로드를 기준으로 작성

```
# 비디오의 어떠한 데이터를 가져올까?
def get_video_data(self):  self: <pytube.api.YouTube object at 0x103635748>
    """Gets the page and extracts out the video data."""
    # Reset the filename incase it was previously set.
    self.title = None
    response = urlopen(self.url)

    if not response:
        raise PytubeError("Unable to open url: {0}".format(self.url))

    html = response.read()
    if isinstance(html, str):
        restriction_pattern = "og:restrictions:age"
    else:
        restriction_pattern = bytes("og:restrictions:age", "utf-8")
```

```
HTTP/1.1 200 OK
Date: Sun, 10 Oct 2010 23:26:07 GMT
Server: Apache/2.2.8 (Ubuntu) mod_ssl/2.2.8 OpenSSL/0.9.8g
Last-Modified: Sun, 26 Sep 2010 22:04:35 GMT
ETag: "45b6-834-49130cc1182c0"
Accept-Ranges: bytes
Content-Length: 13
Connection: close
Content-Type: text/html
```

- 요청 결과가 response 변수에 저장
- 만약 결과가 없으면 에러발생
- 요청 결과를 read() 라는 함수로 읽어옴
  - read(): http통신의 response body를 읽는 함수로 이 youtube 영상의 제약 상황을 읽어와 필요시 로그인을 하도록 유도한다.

다운로드를 기준으로 작성

```
# Extract out the json data from the html response body.
json_object = self._get_json_data(html)
```

```
def _get_json_data(self, html):  self: <pytube.api.YouTube object
    """Extract the json out from the html.

    :param str html:
        The raw html of the page.
    """
    # 18 represents the length of "ytplayer.config = ".
    if isinstance(html, str):
        json_start_pattern = "ytplayer.config = "
    else:
        json_start_pattern = bytes("ytplayer.config = ", "utf-8")
    pattern_idx = html.find(json_start_pattern)
    # In case video is unable to play
    if(pattern_idx == -1):
        raise PytubeError("Unable to find start pattern.")
    start = pattern_idx + 18
    html = html[start:]  html: b'<!DOCTYPE html><html lang="ko" da

    offset = self._get_json_offset(html)
    if not offset:
        raise PytubeError("Unable to extract json.")
    if isinstance(html, str):
        json_content = json.loads(html[:offset])
    else:
        json_content = json.loads(html[:offset].decode("utf-8"))

    return json_content
```

- html에서 json 데이터를 가져온다
- html이 string타입일 경우와 아닌경우로 나누는데 일단 html 페이지를 통채로 읽어 온다.
- 그리고 그것을 json으로 파싱하여 반환함.
- _get_json_offset에서 json으로 변경됨

다운로드를 기준으로 작성

```
def _get_json_offset(self, html):  self: <pytube.api.YouTube ob
    """Find where the json object starts.

    :param str html:
        The raw html of the YouTube page.
    """
    unmatched_brackets_num = 0  unmatched_brackets_num: 1
    index = 1  index: 1
    for idx, ch in enumerate(html):  idx: 9
        if isinstance(ch, int):
            ch = chr(ch)  ch: 116
        if ch == "{":
            unmatched_brackets_num += 1
        elif ch == "}":
            unmatched_brackets_num -= 1
            if unmatched_brackets_num == 0:
                break
    else:
        raise PytubeError("Unable to determine json offset.")
    return index + idx
```

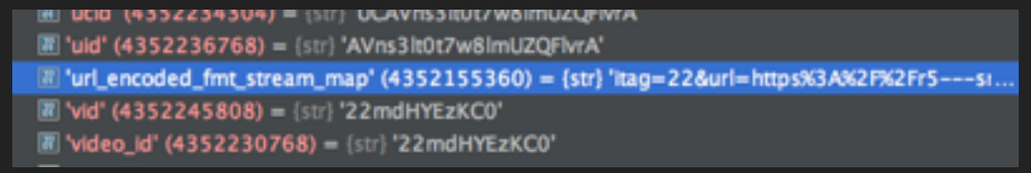
```
b'{"html5":true,"messages":{"player_fallback":["\xeb\x8f\x99\xec\x98\x81\xec\x83\x81\xec\x9d\x84 ↵
$\xec\x9e\xac\xec\x83\x9d\xed\x95\x98\xeb\xa0\xa4\xeb\xa9\xb4 Adobe Flash Player ↵
$\xeb\x98\x90\xeb\x8a\x94 HTML5 \xec\xa7\x80\xec\x9b\x90 ↵
$\xeb\xb8\x8c\xeb\x9d\xbc\xec\x9a\xb0\xec\xa0\x80\xea\xb0\x80 ↵
$\xed\x95\x84\xec\x9a\x94\xed\x95\xa9\xeb\x8b\x88\xeb\x8b\xa4. \\u003ca href=\\"http:\\/\\/get↵
$.adobe.com\\/flashplayer\\/\\"\\u003e\xec\xb5\x9c\xec\x8b\xa0 Flash Player ↵
$\xeb\x8b\xa4\xec\x9a\xb4\xeb\xa1\x9c\xeb\x93\x9c\\u003c\\/a\\u003e \\u003ca ↵
$href=\\"\\/html5\\"\\u003eHTML5 \xeb\xb8\x8c\xeb\x9d\xbc\xec\x9a\xb0\xec\xa0\x80\xeb\xa1\x9c ↵
$\xec\x97\x85\xea\xb7\xb8\xeb\xa0\x88\xec\x9d\xb4\xeb\x93\x9c\xed\x95\x98\xeb\x8a\x94 ↵
$\xeb\xb0\xa9\xeb\xb2\x95 \xec\x9e\x90\xec\x84\xb8\xed\x9e\x88 ↵
$\xec\x95\x8c\xec\x95\x84\xeb\xb3\xb4\xea\xb8\xb0\\u003c\\/a\\u003e"]},"min_version":"8.0.0",↵
$"args":{"atc":"a=3\\u0026b=a4-ulWVlJ7C9IYm7nFEtNJp2QnU\\u0026c=1452911917\\u0026d=1\\u0026e↵
$=22mdHYEzKC0\\u0026c3a=27\\u0026c1a=1\\u0026c6a=1\\u0026hh=uBVaKPVEBoLkXtBW-VXh9PR60XI",↵
$"ad_preroll":"1","cc_load_policy":"1","subtitles_xlb":"https:\\/\\/s.ytimg↵
$.com\\/yts\\/xlbbin\\/subtitles-strings-ko_KR-vfliywS4v.xlb","innertube_api_version":"v1",↵
$"cc3_module":"1","mpvid":"gwyfJ3XQeYoV5F6y","instream":true,↵
```

- html 데이터가 들어오면 {} 체크를 하는데
- unmatched_bracket_num = 0이다가 json데이터의 '{' 가 나오면 1을 더하고 '}'가 나오면 -1을 해서 0이 되면 끝난다.
- 그런데 이 균형이 맞지 않으면 정확한 json 이 아니라고 알려주고
- offset 몇개나 나오는지 반환한다

다운로드를 기준으로 작성

```
def get_video_data(self):  self: <pytube.api.YouTube object at 0x10363b780>
    """Gets the page and extracts out the video data."""
    # Reset the filename incase it was previously set.
    self.title = None
    response = urlopen(self.url)  response: <http.client.HTTPResponse object a

    if not response:
        raise PytubeError("Unable to open url: {0}".format(self.url))

    html = response.read()  html: b'<!DOCTYPE html><html lang="ko" data-cast-a
    if isinstance(html, str):
        restriction_pattern = "og:restrictions:age"
    else:
        restriction_pattern = bytes("og:restrictions:age", "utf-8")  restricti

    if restriction_pattern in html:
        raise AgeRestricted("Age restricted video. Unable to download "
                            "without being signed in.")

    # Extract out the json data from the html response body.
    json_object = self._get_json_data(html)

    # Here we decode the stream map and bundle it into the json object. We
    # do this just so we just can return one object for the video data.
    encoded_stream_map = json_object.get("args", {}).get(
        "url_encoded_fmt_stream_map")
    json_object["args"]["stream_map"] = self._parse_stream_map(
        encoded_stream_map)
    return json_object
```

- html을 json으로 파싱하고,



- stream_map이란걸 가져온다

- 그리고 _parse_stream_map 호출

다운로드를 기준으로 작성

```
def _parse_stream_map(self, blob):  self: <pytube.api.YouTube object at 0
    """A modified version of `urlparse.parse_qs` that's able to decode
    YouTube's stream map.

    :param str blob:
        An encoded blob of text containing the stream map data.
    """
    dct = defaultdict(list)  dct: defaultdict(<class 'list'>, {})

    # Split the comma separated videos.
    videos = blob.split(",")

    # Unquote the characters and split to parameters.
    videos = [video.split("&") for video in videos]

    # Split at the equals sign so we can break this key value pairs and
    # toss it into a dictionary.
    for video in videos:
        for kv in video:
            key, value = kv.split("=")
            dct[key].append(unquote(value))
    log.debug("decoded stream map: %s", dct)
    return dct
```

- 위 메소드에서 호출한 steam_map을 연산하기 좋은 dictionary 형태로 파싱

```
video = {list} ['itag=22', 'url=https%3A%2F%2Fr5---sn-ab02a0nfpgxapox-bh2d.googlevideo.c... View
  __len__ = {int} 5
  0 = {str} 'itag=22'
  1 = {str} 'url=https%3A%2F%2Fr5---sn-ab02a0nfpgxapox-bh2d.googlevideo.com%2Fvideo... View
  2 = {str} 'type=video%2Fmp4%3B+codecs%3D%22avc1.64001F%2C+mp4a.40.2%22'
  3 = {str} 'quality=hd720'
  4 = {str} 'fallback_host=tc.v18.cache8.googlevideo.com'
```

- url 정보를 포함해 type, quality 등을 가져옴.(streaming이 가능한 정보들이 포함) 해상도, 크기, 비디오 타입 등등..

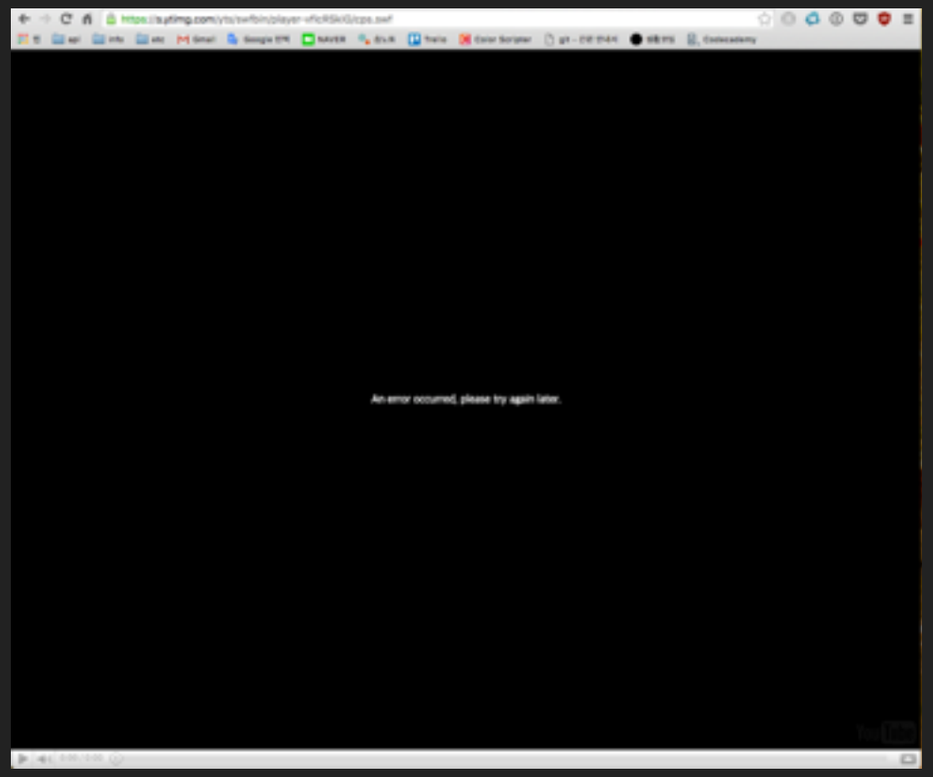
```
'quality' (4350336728) = {list} ['hd720', 'medium', 'medium', 'small']
  __len__ = {int} 4
  0 = {str} 'hd720'
  1 = {str} 'medium'
'type' (4351832896) = {list} ['video/mp4;+codecs="avc1.64001F,+mp4a.40
  __len__ = {int} 5
  0 = {str} 'video/mp4;+codecs="avc1.64001F,+mp4a.40.2"'
  1 = {str} 'video/webm;+codecs="vp8.0,+vorbis"'
  2 = {str} 'video/mp4;+codecs="avc1.42001E,+mp4a.40.2"'
  3 = {str} 'video/x-flv'
  4 = {str} 'video/3gpp;+codecs="mp4v.20.3,+mp4a.40.2"'
```

다운로드를 기준으로 작성

```
# 비디오의 어떠한 데이터를 가져올까?
def get_video_data(self):  self: <pytube.api.YouTube object at 0x10
    """Gets the page and extracts out the video data."""
    # Reset the filename incase it was previously set.
    self.title = None
    response = urlopen(self.url)  response: <http.client.HTTPRespon

    if not response:
        raise PytubeError("Unable to open url: {0}".format(self.url

    html = response.read()  html: b'<!DOCTYPE html><html lang="ko"
    if isinstance(html, str):
        restriction_pattern = "og:restrictions:age"
    else:
        restriction_pattern = bytes("og:restrictions:age", "utf-8")

    if restriction_pattern in html:
        raise AgeRestricted("Age restricted video. Unable to downlo
                            "without being signed in.")

    # Extract out the json data from the html response body.
    json_object = self._get_json_data(html)  json_object: {'html5':

    # Here we decode the stream map and bundle it into the json obj
    # do this just so we just can return one object for the video d
    encoded_stream_map = json_object.get("args", {}).get(  encoded_
        "url_encoded_fmt_stream_map")
    json_object["args"]["stream_map"] = self._parse_stream_map(
        encoded_stream_map)
    return json_object
```

- 이렇게 parsing된 사용자가 요청한 youtube 데이터를 반환해준다

다운로드를 기준으로 작성



- from_url 이란 함수에서 위의 과정을 거친 비디오 data 내용으 이러하다.
- js, css, html5의 내용부터 어떤 내용이 오고 갔는지, url은 무엇인지
- url을 직접 입력해보면 youtube ui가 등장하지만 재생은 되지 않는다.
- youtube 영상 재생시 다른 정보가 더 필요한것 같다.

다운로드를 기준으로 작성

```
# Rewrite and add the url to the javascript file, we'll need to fetch
# this if YouTube doesn't provide us with the signature.
js_url = "http:" + video_data.get("assets", {}).get("js")  js_url: 'http

# Just make these easily accessible as variables.
stream_map = video_data.get("args", {}).get("stream_map")
video_urls = stream_map.get("url")
```

- 이렇게 가져온 data중 javascript 데이터를 찾는다

```
▼ stream_map = {defaultdict} defaultdict(<class 'list'>, {'type': ['video/mp4;+codecs="avc1.6400
    __len__ = {int} 6
  ▶ 'default_factory' (4352121520) = {list} []
  ▶ 'fallback_host' (4352180784) = {list} ['tc.v18.cache8.googlevideo.com', 'tc.v24.cache8.goo
  ▶ 'itag' (4352049984) = {list} ['22', '43', '18', '5', '36', '17']
  ▶ 'quality' (4350336728) = {list} ['hd720', 'medium', 'medium', 'small', 'small', 'small']
  ▶ 'type' (4351832896) = {list} ['video/mp4;+codecs="avc1.64001F,+mp4a.40.2"', 'video/w
  ▼ 'url' (4350336392) = {list} ['https://r5---sn-ab02a0nfpgxapox-bh2d.googlevideo.com/vic
      __len__ = {int} 6
      0 = {str} 'https://r5---sn-ab02a0nfpgxapox-bh2d.googlevideo.com/videoplayback?mt
      1 = {str} 'https://r5---sn-ab02a0nfpgxapox-bh2d.googlevideo.com/videoplayback?mt
      2 = {str} 'https://r5---sn-ab02a0nfpgxapox-bh2d.googlevideo.com/videoplayback?mt
```

- 눈으로 보고 이해할수 있는 내용은
  - quality, type, url
- 그 중에서도 url만 가져온다

다운로드를 기준으로 작성

```
# Check if we have the signature, otherwise we'll need to get the
# cipher from the js.
if "signature=" not in url:
    log.debug("signature not in url, attempting to resolve the "
              "cipher.")
    signature = self._get_cipher(stream_map["s"][idx], js_url)
    url = "{0}&signature={1}".format(url, signature)  url: 'https:/
self._add_video(url, self.filename, **quality_profile)
```

- url에 signature가 있다면
- url에 이 서명을 추가하여 _add_video에 저장한다.
- 그리고 캐쉬를 초기화해준다

다운로드를 기준으로 작성

```
@property
def filename(self):  self: <pytube.api.YouTube object at 0x10363b780>
    """Gets the filename of the video.  If it hasn't been defined by the
    user, the title will instead be used.
    """
    if not self._filename:
        self._filename = safe_filename(self.title)
        log.debug("generated 'safe' filename: %s", self._filename)
    return self._filename
```

- 그리고 filename을 가져오는 과정에서 getter 메소드를 사용하게 되는데 현업에선 아무 조건없이 무조건 가져오는 작업만 했었다. 그런데 여기선 간단한 과정을 거쳐서 filename이 없을경우 어떻게 해야하는지도 작성이 되어있다

```
yt = YouTube("https://www.youtube.com/watch?v=22mdHYEzKC0")
yt.set_filename("paint")
video = yt.get('mp4','720p')
video.download('/Users/yevgnenll/Desktop/')
```

- 이러한 과정을 거쳐 Youtube의 객체가 yt에 생성되었다. 객체 생성과정에서 파일의 이름을 얻었지만 내가 파일 이름을 정의했다.

다운로드를 기준으로 작성

```
def set_filename(self, filename):  self: <pytube.api.YouTube object at 0x1
    """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
```

```
self = {YouTube} <pytube.api.YouTube object at 0x10363b780>
  _filename = {str} 'paint'
  _js_cache = {NoneType} None
  _video_url = {str} 'https://www.youtube.com/watch?v=22mdHYEzKC0'
▶ _videos = {list} [<Video: MPEG-4 Visual (.3gp) - 144p - Simple>, <Video: MPEG-4 Visual (... View
  filename = {str} 'paint'
  title = {str} 'Star Wars in 99 Seconds'
  url = {str} 'https://www.youtube.com/watch?v=22mdHYEzKC0'
  video_id = {str} '22mdHYEzKC0'
▶ videos = {list} [<Video: MPEG-4 Visual (.3gp) - 144p - Simple>, <Video: MPEG-4 Visual (.... View
```

- 파일이름을 셋팅하는 과정이다.
- setter method를 사용할때 보통 return 값을 정의하지 않았는데, 여기서는 True를 정의했다. 이거 다음에 나도 써먹어야겠다!
- 파일이름, cashe정보, video url, 영상 원래의 제목,

다운로드를 기준으로 작성

```
def get(self, extension=None, resolution=None, profile=None):  self: <pytube.
    """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 profile (this is subjective, I don't 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.")
```

```
video = yt.get('mp4','720p')
```

- 내 코드엔 파일 확장자명, 해상도만 입력했지만 원래는 quality도 입력이 가능하다
- 하지만 pytube를 만든 프로그래머는 이 옵션을 사용하지 않을것을 권장한다.
- Youtube의 객체 생성시 사용했던 메소드 get_videos()에서 정보를 가져오는데 내가 입력한 옵션이 해당하는지를 확인한다.
- 이 결과를 video에 저장한다.

```
yt = YouTube("https://www.youtube.com/watch?v=22mdHYEzKC0")
yt.set_filename("paint")
video = yt.get('mp4','720p')
video.download('/Users/yevgnenll/Desktop/')
```

다운로드를 기준으로 작성

```
video = {Video} <Video: H.264 (.mp4) - 720p - High>
  audio_bitrate = {str} '192'
  audio_codec = {str} 'AAC'
  extension = {str} 'mp4'
  filename = {str} 'paint'
  profile = {str} 'High'
  resolution = {str} '720p'
  url = {str} 'https://r5---sn-ab02a0nfpgxapox-bh2d.googlevideo.com/videoplayback?mt=1..
  video_bitrate = {str} '2-2.9'
  video_codec = {str} 'H.264'
```

- video에 저장된 내용

```
yt = YouTube("https://www.youtube.com/watch?v=22mdHYEzKC0")
yt.set_filename("paint")
video = yt.get('mp4','720p')
video.download('/Users/yevgnenll/Desktop/')
```

- download 시작

MODEL.PY, DOWNLOAD

```
def download(self, path, chunk_size=8 * 1024, on_progress=None,  self: <Video: H.264 (.mp4) - 720p - High>  chunk_size: 8192  on_progress: None
             on_finish=None, force_overwrite=False):  on_finish: None  force_overwrite: False
    """Downloads the video.

    :param str path:
        The destination output directory.
    :param int chunk_size:
        File size (in bytes) to write to buffer at a time. By default,
        this is set to 8 bytes.
    :param func on_progress:
        *Optional* function to be called every time the buffer is written
        to. Arguments passed are the bytes recieved, file s
        datetime.
    :param func on_finish:
        *Optional* callback function when download is compl
        passed are the full path to downloaded the file.
    :param bool force_overwrite:
        *Optional* force a file overwrite if conflicting or
    """
```

```
self = {Video} <Video: H.264 (.mp4) - 720p - High>
  audio_bitrate = {str} '192'
  audio_codec = {str} 'AAC'
  extension = {str} 'mp4'
  filename = {str} 'paint'
  profile = {str} 'High'
  resolution = {str} '720p'
  url = {str} 'https://r5---sn-ab02a0nfpgxapox-bh2d.googlevideo.com/videoplayback?requi...
  video_bitrate = {str} '2-2.9'
  video_codec = {str} 'H.264'
```

지금 입력한내용은 다운로드 받을 위치만 보냈다.

이미 전송된 video 데이터에, 다운로드 경로로 다운로드를 시작한다.

테스트시 파일이 이미 경로에 있다는 에러가 나온적이 있는데 force_overwrite에 True를 입력하면

이 에러가 발생하지 않는다.

다운로드를 기준으로 작성

```
path = os.path.normpath(path)
if os.path.isdir(path):
    filename = "{0}.{1}".format(self.filename, self.extension)
    path = os.path.join(path, filename)  path: '/Users/yevgnenll/Desktop/'
# 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))
```

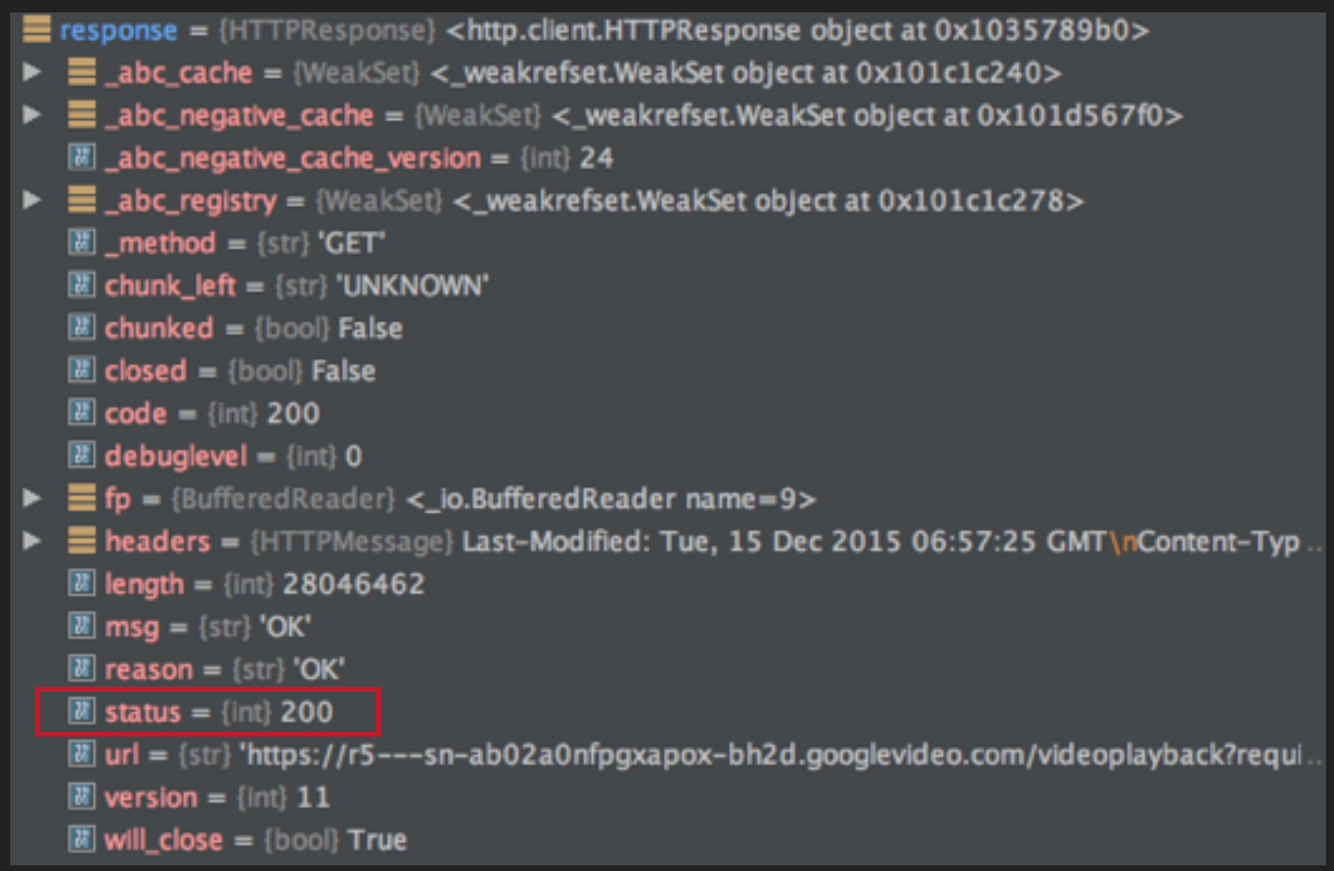
- path, 경로가 os마다 조금씩 다르게 나오는데 이것을 정규화 해준다.
- 그리고 path와 파일이름, 확장자를 합쳐서 최종경로와 파일이름을 붙여놓는다 join 함수 사용.

```
# 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.
```

- 파일이 이미 경로에 있고, 덮어쓰기가 False 라면 에러를 발생시킨다

다운로드를 기준으로 작성

```
# 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"))
```



- urlopen 함수로 해당 youtube 영상에 접근하고
- response에 http통신 이후의 결과를 저장한다.

response에 저장된 내용은 이러하다.

- cache, method방식(get), 접속 여부등, 다운로드 할때 stream을 어떻게 처리하는지 등의 http의 내용이 들어있다
TCP/IP 꼭 읽어봐야할듯

- 연결 완료 상태

다운로드를 기준으로 작성

```
# TODO: Split up the downloading and OS jazz into separate functions.
response = urlopen(self.url)  response: <http.client.HTTPResponse object at 0x1035789b0>
meta_data = dict(response.info().items())  meta_data: {'Connection': 'close', 'Server': '
file_size = int(meta_data.get(                                    ) or
                meta_data.get("content-length"))
```

```
meta_data = {dict} {'Connection': 'close', 'Server': 'gvs 1.0', 'Accept-Ranges': 'b
 __len__ = {int} 11
 'Accept-Ranges' (4351070000) = {str} 'bytes'
 'Alternate-Protocol' (4351064728) = {str} '443:quic,p=.45'
 'Cache-Control' (4351040048) = {str} 'private,
 'Connection' (4351069616) = {str} 'close'
 'Content-Length' (4351069680) = {str} '28046462'
 'Content-Type' (4351038896) = {str} 'video/mp4'
 'Date' (4351035296) = {str} 'Sat, 16 Jan 2016 14:53:42 GMT'
 'Expires' (4351035576) = {str} 'Sat, 16 Jan 2016 14:53:42 GMT'
 'Last-Modified' (4351040240) = {str} 'Tue, 15 Dec 2015 06:57:25 GMT'
 'Server' (4351035912) = {str} 'gvs 1.0'
 'X-Content-Type-Options' (4351065304) = {str} 'nosniff'
```

비디오 용량

- meta_data엔 어떤 내용이 저장될까?
- metadata(위키): 다른 데이터를 설명해주는 데이터로 일정한 규칙에 따라 컨텐츠에 부여되는 데이터. 어떤 구조화된 정보를 분석, 분류하고 부가적인 정보를 추가하기 위해서 데이터 뒤에 함께 따라가는 정보를 말한다.
- 사진의 경우 촬영당시 시간, 노출, 플래시 사용여부, 해상도, 사진크기 등의 정보 필요시 gps 정보 저장도 가능하다

다운로드를 기준으로 작성

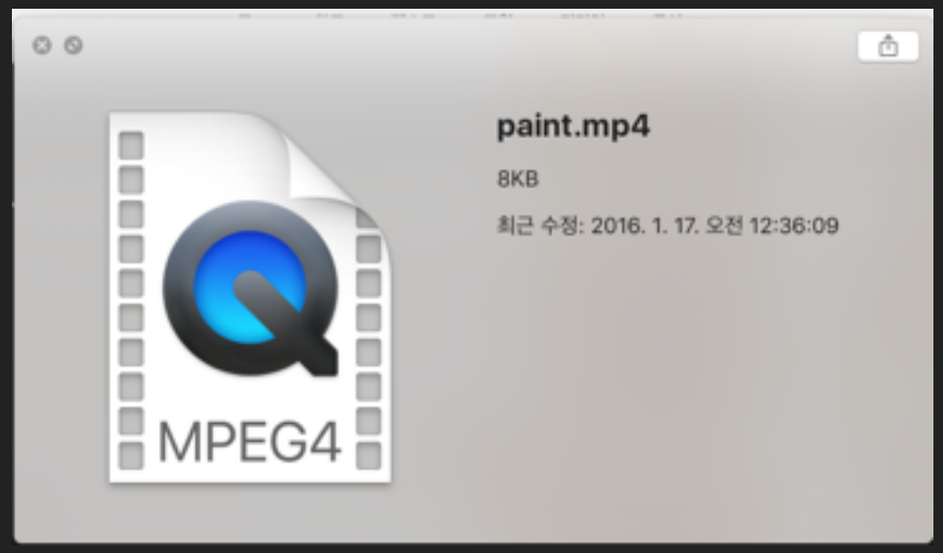
```
file_size = int(meta_data.get("Content-Length") or  file_size: 28046462
                meta_data.get("content-length"))
```

- 메타데이터에서 파일의 사이즈를 찾아옴
- Content_length와 content_Length 두가지로 검색을 하고 결과를 받아온다.

```
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:
                    # TODO: We possibly want to flush the
                    # `_bytes_recieved`` buffer before we call
                    # ``on_finish()``.
                    on_finish(path)
                break

            self._bytes_received += len(self._buffer)
            dst_file.write(self._buffer)
            if on_progress:
                on_progress(self._bytes_received, file_size, start)
```

- 실제적으로 파일쓰기를 시작한다.
- http 통신의 응답을 읽고(특별히 지정하지 않으면, 8kbyte씩 읽어내려간다.)
- 목적경로에 파일을 쓰기 시작한다

다운로드를 기준으로 작성



- while문을 딱 한번 돌았을때의 paint.mp4 파일의 상태.

- while문이 모두 돌고나면 이렇게 동영상 다운로드가 완료된다.

다운로드를 기준으로 작성

```
except KeyboardInterrupt:
    # TODO: Move this into the cli, ``KeyboardInterrupt`` handling
    # should be taken care of by the client. Also you should be allowed
    # to disable this.
    os.remove(path)
    raise KeyboardInterrupt(
        "Interrupt signal given. Deleting incomplete video.")
```

- 에러가 나거나 취소가 되면 exception처리를 한다.
- 이때 os.remove(path)는 해당 경로에 다운 받고 있는 동영상 파일을 삭제한다
- 바로 전 슬라이드에서 8kb만큼 다운이 되었었는데 중간에 디버그 모드를 종료하면 동영상 파일은 자동 삭제가 되었다.

텍스트

```
class YouTube(object):
    """Class representation of a single instance of a YouTube session.
    """
    def __init__(self, url=None):...

    @property
    def url(self):...

    @url.setter
    def url(self, url):...

    @property
    def video_id(self):...

    @property
    def filename(self):...

    @filename.setter
    def filename(self, filename):...

    def set_filename(self, filename):...

    def get_videos(self):...

    @property
    def videos(self):...

    def from_url(self, url):...

    def get(self, extension=None, resolution=None, profile=None):...

    def filter(self, extension=None, resolution=None, profile=None):...

    # 비디오의 어떠한 데이터를 가져올까?
    def get_video_data(self):...

    def _parse_stream_map(self, blob):...

    def _get_json_data(self, html):...

    def _get_json_offset(self, html):...

    def _get_cipher(self, signature, url):...

    def _get_quality_profile_from_url(self, video_url):...

    def _add_video(self, url, filename, **kwargs):...
```

```
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):...

    def download(self, path, chunk_size=8 * 1024, on_progress=None,
                 on_finish=None, force_overwrite=False):...

    def __repr__(self):...

    def __lt__(self, other):...
```

텍스트

```
class YouTube(object):
    """Class representation of a single instance of a YouTube session.
    """
    def __init__(self, url=None):...

    @property
    def url(self):...

    @url.setter
    def url(self, url):...

    @property
    def video_id(self):...

    @property
    def filename(self):...

    @filename.setter
    def filename(self, filename):...

    def set_filename(self, filename):...

    def get_videos(self):...

    @property
    def videos(self):...

    def from_url(self, url):...

    def get(self, extension=None, resolution=None, profile=None):...

    def filter(self, extension=None, resolution=None, profile=None):...

    # 비디오의 상태를 얻어오는 방법은?
    def get_video_data(self):...

    def _parse_stream_map(self, blob):...

    def _get_json_data(self, html):...

    def _get_json_offset(self, html):...

    def _get_cipher(self, signature, url):...

    def _get_quality_profile_from_url(self, video_url):...

    def _add_video(self, url, filename, **kwargs):...
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14

```
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):...

    def download(self, path, chunk_size=8 * 1024, on_progress=None,
                 on_finish=None, force_overwrite=False):...

    def __repr__(self):...

    def __lt__(self, other):...
```

텍스트

```
class YouTube(object):
    """Class representation of a single instance of a YouTube session.
    """
    def __init__(self, url=None):...

    @property
    def url(self):...

    @url.setter
    def url(self, url):...

    @property
    def video_id(self):...

    @property
    def filename(self):...

    @filename.setter
    def filename(self, filename):...

    def set_filename(self, filename):...

    def get_videos(self):...

    @property
    def videos(self):...

    def from_url(self, url):...

    def get(self, extension=None, resolution=None, profile=None):...

    def filter(self, extension=None, resolution=None, profile=None):...

    # 비디오의 어떠한 데이터를 가져올까?
    def get_video_data(self):...

    def _parse_stream_map(self, blob):...

    def _get_json_data(self, html):...

    def _get_json_offset(self, html):...

    def _get_cipher(self, signature, url):...

    def _get_quality_profile_from_url(self, video_url):...

    def _add_video(self, url, filename, **kwargs):...
```

- video_id: data가 hash형태로 반환되는데 문자열 쿼리를 파싱한다
- videos: deprecated. user get_videos
- _get_cipher: 유튜브에서 동영상에 있는 서명을 가져온다(암호화 되었을때 가져오는듯, JSinterpreter가 이때 사용된다)
  cipher: 암호
- _get_quality_profile_from_url
  get() 에서 비디오 정보를 가져올때 profile까지 요청할때가 있다(high 같은 동영상의 어떠한 품질) 하지만 pytube 작성자는 이 항목의 사용을 비추함.