Pytube

Api.py
한민수

실행 화면에서 직접적으로 쓰이는 코드를 살펴보자

```
from pytube import YouTube
yt = YouTube("https://www.youtube.com/watch?v=EWzkmRV_Jgs")
yt.set_filename('afa')
video =yt.get('mp4','240p')
video.download('C:\\Users\\hanminsoo\\PycharmProjects')
```

1. Pytube 패키지의 Youtube 클래스를 import 합니다.
2. Youtube 클래스를 호출하여 url값을 넣어줍니다.
3. Set_filename메소드를 호출하여 file이름 값을 넣어줍니다.
4. get메소드에 파일 '확장자'와 '해상도' 값을 넣어줍니다.
5. download 메소드를 호출하여 해당 링크에 영상을 다운로드합니다.

실행 화면에서 직접적으로 쓰이는 코드를 살펴보자

-api.py

```
class YouTube(object):
    """Class representation of a single instance of a YouTube session.
    """
    def set_filename(self, filename):...

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

1. Youtube클래스로 url이 매개변수로 입력됩니다.
2. set_filename, get메소드가 여기 있었습니다.

처음부터 차례차례...

-url을 입력받는 것부터

```
class YouTube(object):#youtube의 출처를 받는 클래스입니다. youtube함수를 호출받아서 경로를 지정받으면...
    """Class representation of a single instance of a YouTube session.
    """
    def __init__(self, url=None):
        """Initializes YouTube API wrapper...."""
        self._filename = None
        self._video_url = None
        self._js_cache = None
        self._videos = []
        if url:#경로를 지정받았으므로 from_url을 호출하여(url 인자를 넘겨줍니다)
            self.from_url(url)
```

1. Youtube에 url을 입력받습니다.
2. 생성자를 통해서 변수를를 초기화 시켜주고 조건문을 통해 **from_url**을 호출합니다.

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

    :param str url:
        The url to the YouTube video.
    """
    self._video_url = url
    self._filename = None
    self._videos = []

    video_data = self.get_video_data()

    self.title = video_data.get("args", {}).get("title")

    js_url = "http:" + video_data.get("assets", {}).get("js")

    stream_map = video_data.get("args", {}).get("stream_map")
    video_urls = stream_map.get("url")

    for idx, url in enumerate(video_urls):
        log.debug("attempting to get quality profile from url: %s", url)
        try:
            itag, quality_profile = self._get_quality_profile_from_url(url)
            if not quality_profile:
                log.warn("unable to identify profile for itag=%s", itag)
                continue
        except (TypeError, KeyError) as e:
            log.exception("passing on exception %s", e)
            continue

        if "signature=" not in url:
            log.debug("signature not in url, attempting to resolve the "
                      "cipher.")
            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)
        self._add_video(url, self.filename, **quality_profile)

    self._js_cache = None
```

1. url을 저장하고 filename과 videos 초기화.
2. Video_data는 get_video_data메소드를 호출하여 최종 데이터를 저장합니다.

```
def get_video_data(self):#비디오 상세 정보들을 가져옴
    """Gets the page and extracts out the video data."""
    # Reset the filename incase it was previously set.타이틀 초기화
    self.title = None
    response = urlopen(self.url)#urlopen 메소드 호출(urllib모듈에서 호출)

    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")

    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)#html의 json값들을 디코딩해서 가져옵니다

    # 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
```

1. 먼저 title값을 초기화

2. urlopen 메소드 호출(urllib모듈에서 호출)
   urlopen은 웹페이지 없이 해당 해당 url의
   웹사이트를 읽어오는 메소드입니다

3. response를 바이트 문자열로 읽어옵니다(read)

4. isinstance(html,str)현재 사용하고 있는
   데이터 타입의 일치 여부를 확인하는 메소드입니다

5. bytes()메소드는 자료구조를 바이트 배열로
   만들어주는 역할을 합니다(주소값으로 저장됨).

6. html에 restriction_patter값이 있으면 나이 제한
   때문에 다운 받을수 없다는 예외가 발생합니다.

7. html의 json값들을 디코딩해서 가져옵니다.

8. encoded_stream_map에 args키와
   url_encoded_fmt_stream_map키에 저장된 value값을
   가져옵니다.
   (args키 값에 value가 없다면 {}를 가져옴).

```python
def _get_json_data(self, html):
    """Extract the json out from the html.

    :param str html:
        The raw html of the page.
    """
    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    "Ytplayer.config = "바로 이 부분"
    html = html[start:]

    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
```

1. html의 데이터 타입이 string일 경우와 그렇지 않을 경우를 나눠 json_start_pattern에 "ytplayer.config=" 라는 문자열을 저장합니다.

2. find메소드는 데이터 안에 해당 문자열이 몇 번째에 있는지 알려주는 역할을 합니다.
결국 pattern_idx에는 html에 ytplayer.config = 가 시작되는 부분의 숫자값이 들어있는 셈입니다.

3. Offset은 ytplayer.config = 값의 다음 부분부터 저장된 리스트의 길이를 받아옵니다.

4. 조건물을 통해서 받아온 덤프된 jason값을 받아옵니다. 만약 받아온 값이 str형 이 아닐경우 utf-8방식을 통해서 디코딩합니다.

5. 최종적으로json_content은 html의 "ytplayer.config =" 문자열이 끝나는 부분~끝 까지 저장되어 있는 값을 디코딩하거나 str값으로 저장합니다

(html은 url에서 "ytplayer.config = "까지의 값이 제외된 값들이 저장되어있습니다)

```python
def _get_json_offset(self, html):
    """Find where the json object starts.
    #json object의 시작이 어디인지 찾아줍니다.
    :param str html:
        The raw html of the YouTube page.
    """
    unmatched_brackets_num = 0
    index = 1
    for idx, ch in enumerate(html):
        if isinstance(ch, int):
            ch = chr(ch)
        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
```

1. enumerate함수는 해당 매개변수(자료구조가)의 각 인덱스와 결과값을 한번에 반환합니다.
   예를들어 ['민수','철수']이라면
   0민수, 1철수 값이 한번에 반환된다는 뜻!!

2. ch의 데이터 타입이 정수형 타입일 경우 charecter형으로 바꿔줍니다.

3. Unmatched_brackets_num은 {}의 짝을 맞춰주기 위해서 만들어 놓은 변수였구나!

4. for문에 else, 반복도중 break를 만나지 않는다면 마지막에 무조건 else로 간다 break를 만나면 else를 거치지 않는다

5. 최종적으로 index + idx는 html의 "ytplayer.config =" 문자열의 다음 부분부터 저장된 리스트의 길이를 뜻한다.

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

    :param str url:
        The url to the YouTube video.
    """
    self._video_url = url
    self._filename = None
    self._videos = []

    video_data = self.get_video_data()

    self.title = video_data.get("args", {}).get("title")

    js_url = "http:" + video_data.get("assets", {}).get("js")

    stream_map = video_data.get("args", {}).get("stream_map")
    video_urls = stream_map.get("url")

    for idx, url in enumerate(video_urls):
        log.debug("attempting to get quality profile from url: %s", url)
        try:
            itag, quality_profile = self._get_quality_profile_from_url(url)
            if not quality_profile:
                log.warn("unable to identify profile for itag=%s", itag)
                continue
        except (TypeError, KeyError) as e:
            log.exception("passing on exception %s", e)
            continue

        if "signature=" not in url:
            log.debug("signature not in url, attempting to resolve the "
                      "cipher.")
            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)
        self._add_video(url, self.filename, **quality_profile)

    self._js_cache = None
```

1. 반환 받은 값에서 args키 속의 title키 속에 있는 value를 title로 입력해 줍니다.
2. 같은 방법으로 assets키 속의 js키의 값을 찾아 js_url에 저장해 줍니다.(이름으로 보아 javascript의 url인 것 같습니다)
3. Video_data를 통해서 stream_map과 video_urls를 저장합니다.
4. Enumerate메소드를 통해서 각 조건하에 log를 찍습니다.
5. Url에 "signature="의 문자열이 없을 경우 _get_cipher 메소드를 호출합니다.
6. Url에 signature=부분을 찾아 앞뒤로 url과 signature를 덧씌웁니다.
7. 이렇게 "_video_ur"l과 "비디오의 상세 데이터"를 뽑아 왔습니다.

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

    reg_exp = re.compile(r'₩.sig₩|₩|([a-zA-Z0-9$]+)₩(')

    if not self._js_cache:
        response = urlopen(url)
        if not response:
            raise PytubeError("Unable to open url: {0}".format(self.url))
        self._js_cache = response.read()
    try:
        matches = reg_exp.search(self._js_cache)
        if matches:
            # Return the first matching group.
            func = next(g for g in matches.groups() if g is not None)
        # Load js into JS Python interpreter.
        jsi = JSInterpreter(self._js_cache)
        initial_function = jsi.extract_function(func)
        return initial_function([signature])

    except Exception as e:
        raise CipherError("Couldn't cipher the signature. Maybe YouTube "
                          "has changed the cipher algorithm. Notify this "
                          "issue on GitHub: {0}".format(e))
    return False
```

1. '₩.sig₩|₩|'의 조건을 만족하는 문자열중에 (a-z, A-Z, 0-9)로 끝나는 문자열만 골라서 출력 #r은 raw String의 법칙!!
2. 현재 _js_cache는 None값이 들어있습니다..
3. js_url을 response에 입력 한 후에 조건에 따라 예외 상황을 발생 시킵니다.
4. 그 후 response를 읽어와 _js_cache에 저장합니다.

Set_filename에서는 무엇을 리턴하나?

```python
def set_filename(self, filename): #파일이름을 받습니다.
    """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 #self._filename에 filename출력
    if self.get_videos():
        for video in self.get_videos():
            video.filename = filename
    return True
```

변수 값이 없을 경우 safe_filename메소드를 호출하여 타이틀 값으로

```python
@property
def filename(self):#파일이름을 설정합니다. 만약 사용자에 의해 정의되지 않았다면 타이틀이 사용될것입니다.
    """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:#파일이름이 없으면
        self._filename = safe_filename(self.title)#파일 타이틀을 이름으로 합니다.
        log.debug("generated 'safe' filename: %s", self._filename)
    return self._filename #파일 이름을 리턴합니다.
```

```python
@filename.setter
def filename(self, filename):
    """Sets the filename (This method is deprecated. Use `set_filename()`
    instead).

    :param str filename:
        The filename of the video.
    """
    warnings.warn("filename setter deprecated. Use `set_filename()` "
                  "instead.", DeprecationWarning)
    self.set_filename(filename)
```

변수에 접근해서 변경해줌

1. 생성될 동영상 파일의 이름을 받아옵니다.
2. 파일 이름을 받은 후 조건문으로 get_videos()메소드 호출
3. ** 이렇게 filename을 설정했습니다

get 메소드는 무슨 역할을 할까?

```
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 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.")
```

1. Pytube를 실행 시킬때 입력했던 해상도 값, 확장자 값을 입력 합니다.
   디폴트값으로 profile이라는 값도 있는데 권장하지 않는다고 나와 있습니다.

2. get_videos() 메소드에서 값을 리턴 받아 초기화 되어있는지 확인합니다.
   값이 있을 경우 result 리스트에 넣습니다.

3. 리스트의 수를 받아서 조건문으로 매치되는 확장자,해상도,프로파일을 확인합니다.
   0개 매치될 경우 -> 동영상이 없다
   1개 매치될 경우 -> 확장자값을 리턴합니다.
   2개 이상 매치될 경우 -> 비디오가 여러 개 있습니다.

4. 매치된 확장자 값을 video에 저장합니다.